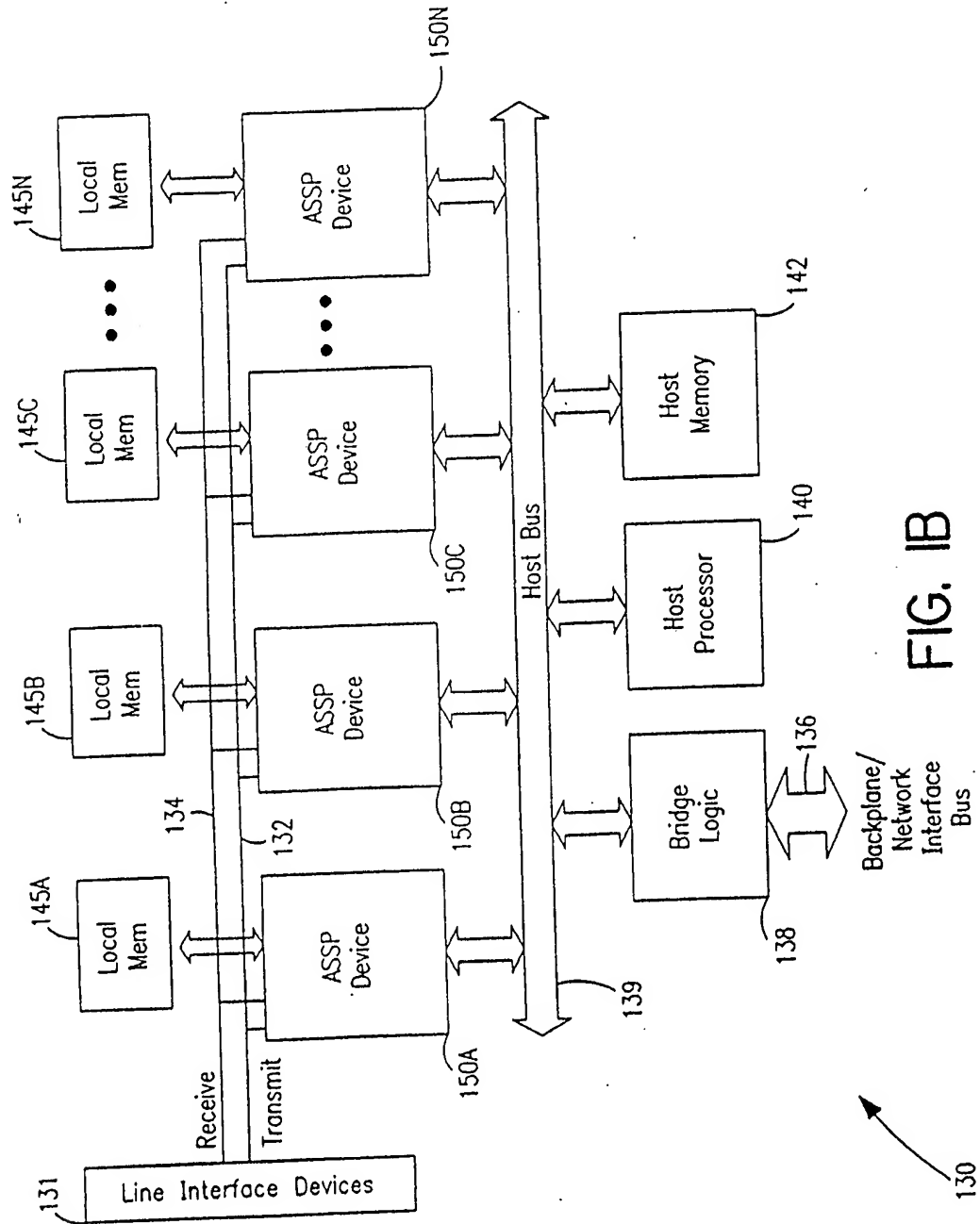


FIG. 1A



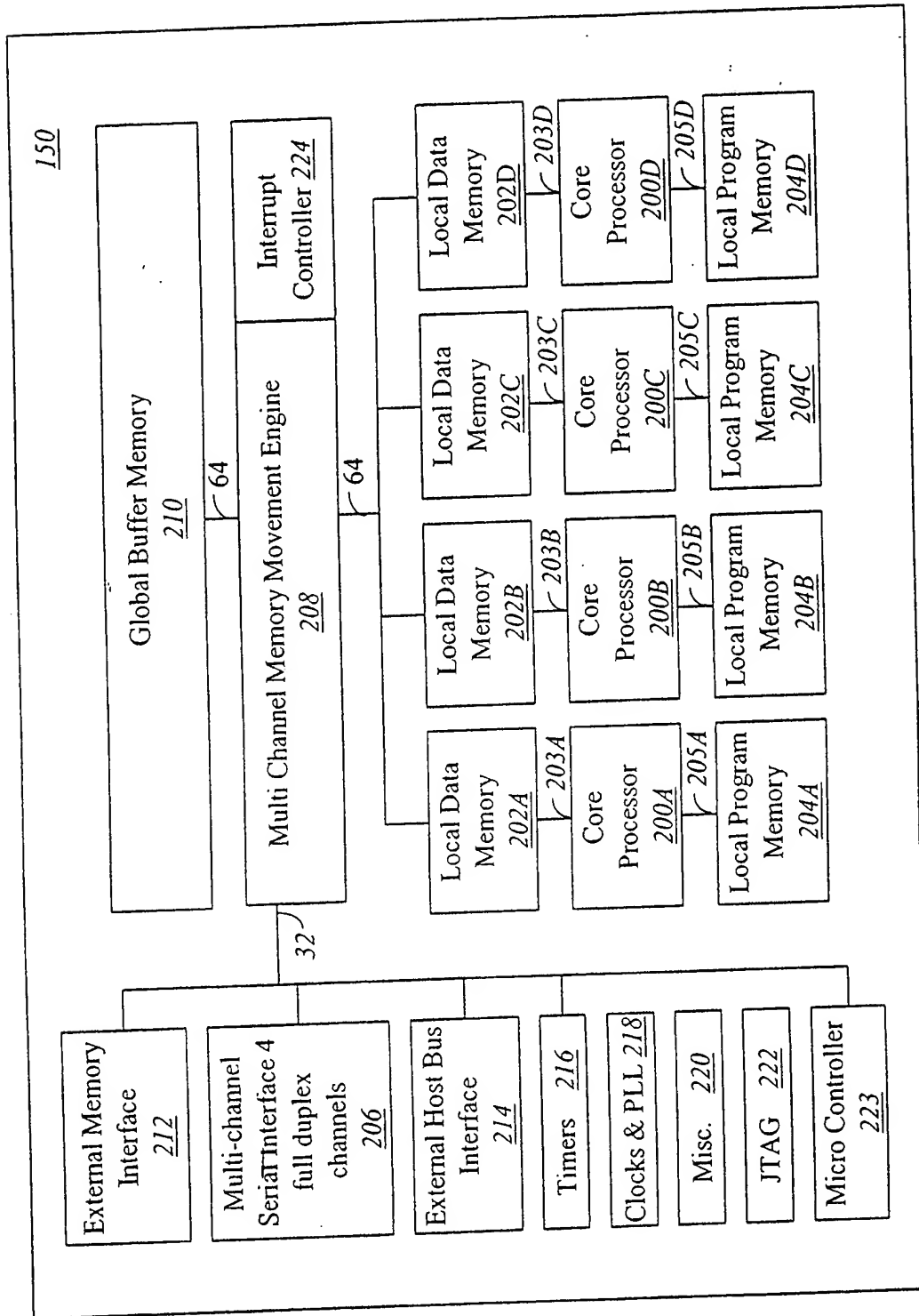


FIG. 2

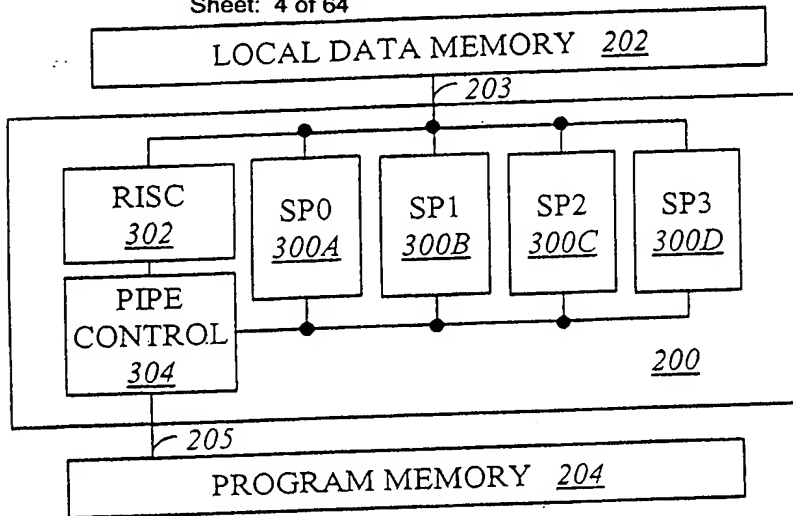


FIG. 3

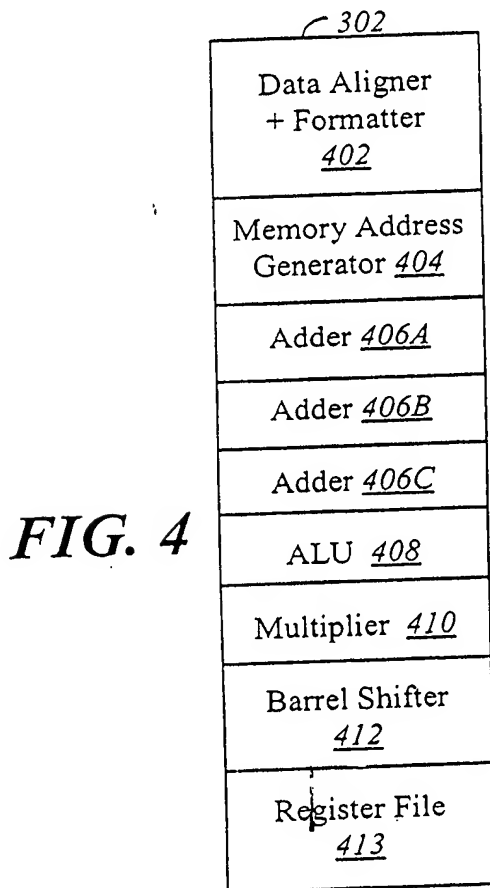


FIG. 4

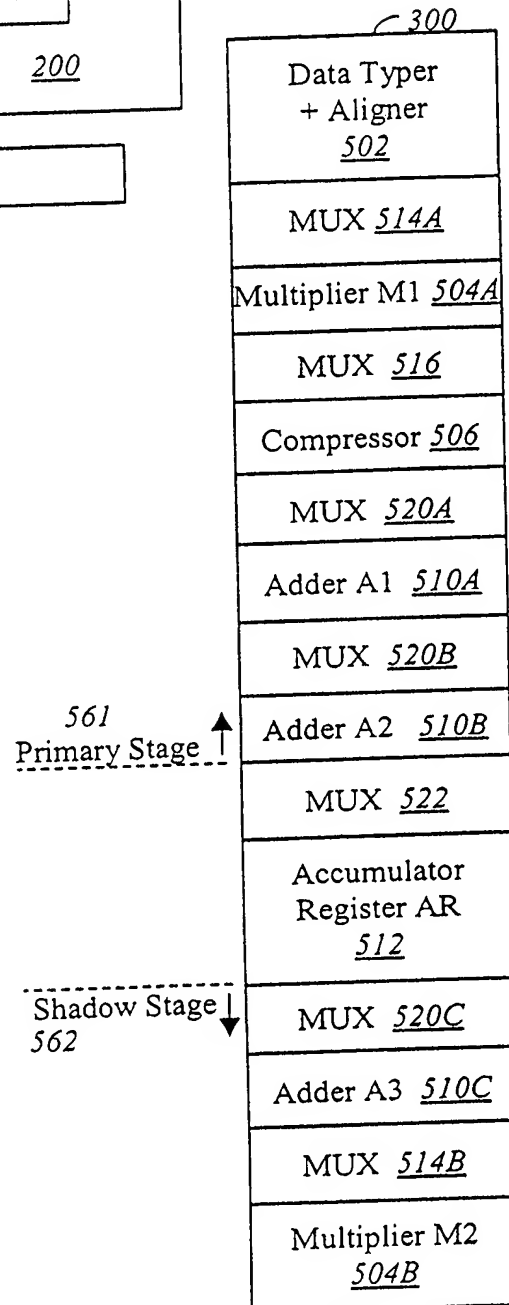


FIG. 5A

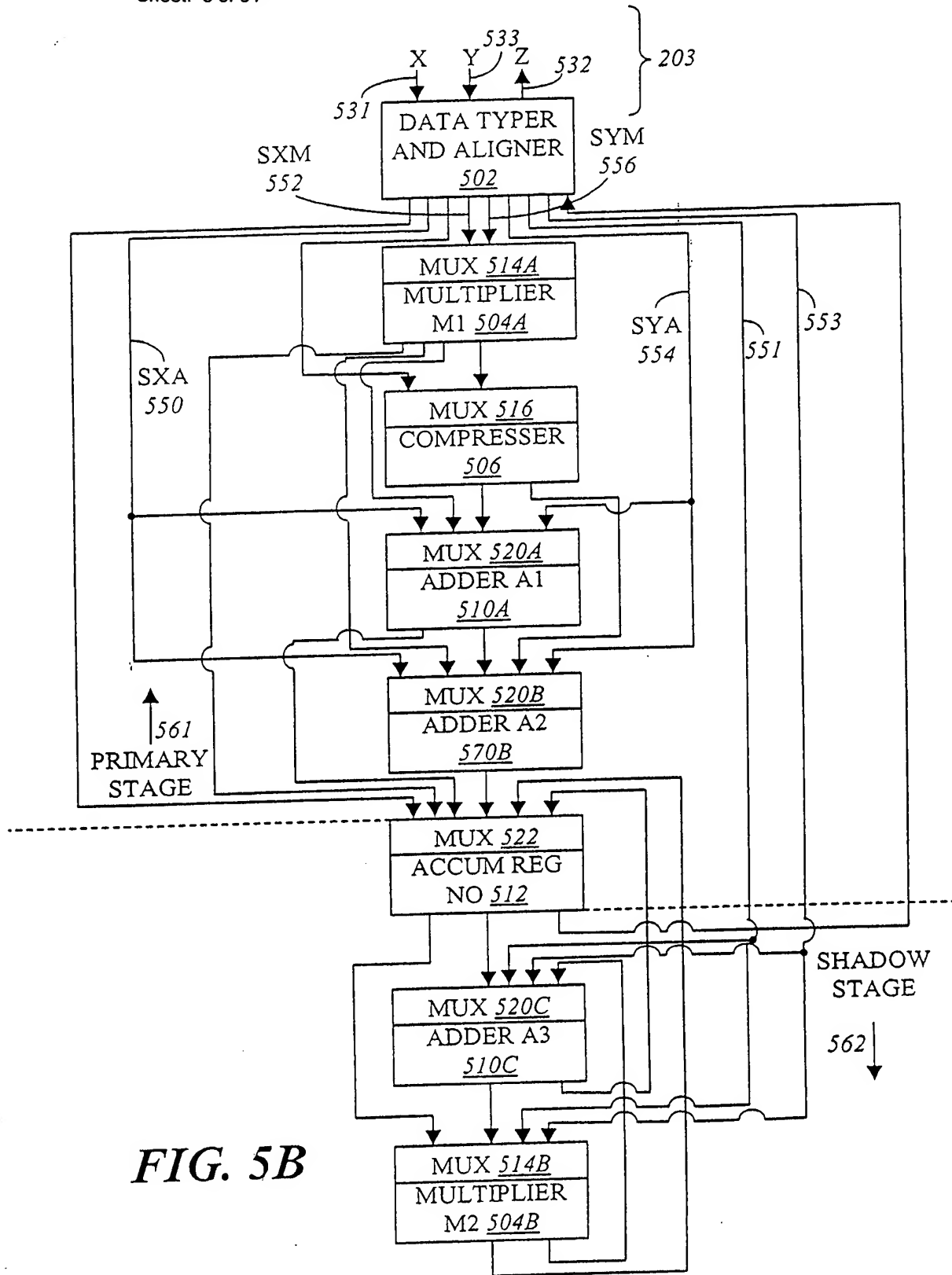


FIG. 5B

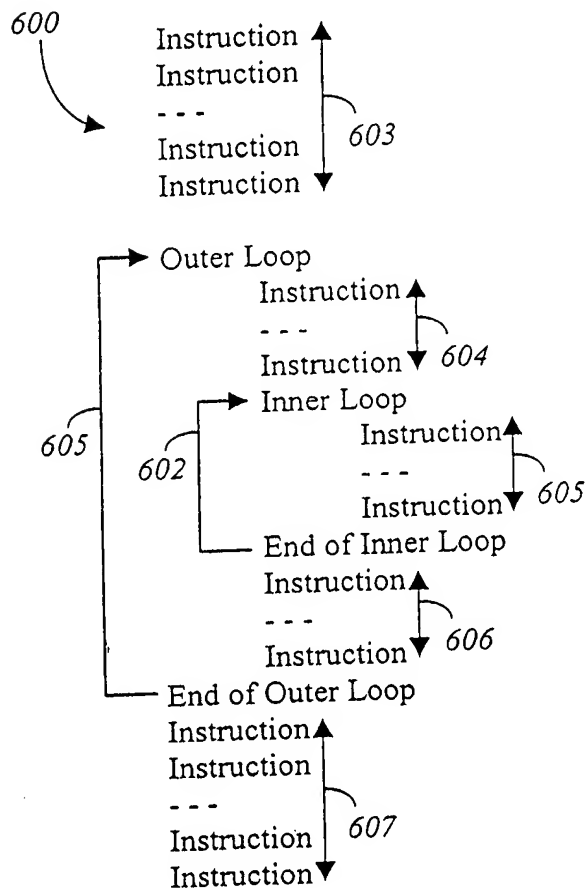


FIG. 6A

610	
611 MAIN OP	612 SUB OP
MULT	NOP
ADD	MIN/MAX
MIN/MAX	ADD
NOP	MULT

FIG. 6G

20-bit ISA	39	19	20-bit parallel	Control II Control
	10	0		
	0	1		
	1	0		
	1	1		
			20-bit serial	Control # Control
			40-bit extended	DSP, extensions/Shadow
			20-bit serial	DSP # DSP

FIG. 6B

6-bit operand specifier:

A 6-bit specifier is used in DSP extended instructions to access memory and register operands.

5	4	3	2	1	0
---	---	---	---	---	---

M/R

0	0	ac-page	ereg
0	1	gpr:0-r15	GPR
1	ptr(r0)to(r15)	off	Mem[ptr{0-15}]II PTR[0-15]+=offset1/offset2 Always postupdate

This allows access to data memory, ereg and GPR

- Bit 5=1:Use rX(X:0-7) register to obtain effective memory address and post-modify the

ptr field by one of two possible offsets specified in rX registers.

dmem[ptr], ptr=ptr+offset1, if off=0

ptr=ptr+offset2, if off=1

- Bit 5=0:Access ac-page or GPR

If Bit-4 is set to 0, then bits 3:0 control access to the general-purpose register file (r0-15) or to execution unit registers.

GPR	GPR Intr page	ac-page	ac intr page	ereg-Shadow DSP
R0	R0	A0	A0 I	A0
R1	R1	A1	A1 I	A1
R2	R2	T	T	T
R3	R3	TR	TR	TR
R4	R4			
R5	R5			
R6	R6			
R7	R7			
R8	R8			SX1
R9	R9			SX1s
R10	R10			SX2
R11	R11			SX2s
R12	R12 i			SY1
R13	R13 i			SY1s
R14	R14 i			SY2
R15	R15 i			SY2s

FIG. 6C

4-bit operand specifier:

Memory operands: (rX) specifies an access out of the data memory to the extension unit for the function that needs to be performed. The address for the access is specified in the rX register in the general register file that hold the 14-bit pointer (16K of addressing) to memory, 5-bit signed offset or a 3-bit unsigned offset that can post-modify the address. In addition, each pointer is typed for efficient SIMD processing and includes a permute control for rearranging data elements of a vector on the fly. The "podi" core can deal with 4-element 16-bit real vectors or complex data directly. This ability to manipulate memory data directly reduces the instruction width greatly and allows efficient signal processing.

(rX): Memory Address Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
type				cb	x	permute				off1:(0-7)				off0:(-16 to 15)				ptr: pointer													

Fig. 6D

For shadow DSP instructions, the 3-bit specifier for operands is defined as follows:

2	1	0	
0	0	0	A0
0	0	1	A1
0	1	0	T
0	1	1	TR
1	0	0	SX1
1	0	1	SX1s
1	1	0	SX2
1	1	1	SX2s
EREG1			

2	1	0	
0	0	0	A0
0	0	1	A1
0	1	0	T
0	1	1	TR
1	0	0	SY1
1	0	1	SY1s
1	1	0	SY2
1	1	1	SY2s
EREG2			

Only the shadow DSP instructions can see the above modified page of execution unit registers.

FIG. 6E

5-bit operand specifier:

The 5-bit specifier includes the 4-bit specifier for general data operands and the special purpose registers. It is used in RISC instructions.

4	3	2	1	0
0	spr:s0-s15			
1	gpr:r0-r15			

SPR		Intr page	
	SPR	SPR	intr page
0	fu-ctl	fu-ctl_l	
1	a-type	a-type_l	
2	ps-ctl	ps-ctl	
3	t-type	t-type	
4	pl-ctl	pl-ctl	
5	cb-ctl	cb-ctl_l	
6	shuffle	shuffle	
7	lo-ptr	lo-ptr	
8	status	status_l	
9	loop-ctl	loop-ctl	
10	pcr	pcr	stack(8)
11	reserved	reserved	
12	reserved	reserved	
13	reserved	reserved	
14	reserved	reserved	

NOTE: All SPR registers are reset to all zeros at power on reset except for the PCR register.

FIG. 6F

DSP instructions

39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20																				
Multiply	1	0	0	PS	S'	SX				SY				N/S	SA	DA	Sub-op			
	da=sx*sy																0	0	0	Nop
	da=(sx*sy)+sa																0	0	1	Add
	da=(sx*sa)+sy																0	1	0	Add
	da=(sx*sy)+sa																0	1	1	Sub
	da=(sx*sa)+sy																1	0	0	Sub
	da=min(sx*sy,sa)																1	0	1	Min
	da=min(sx*sa,sy)																1	1	0	Min
Add	1	0	1	PS	+/-	SX				SY				N/S	SA	DA	Sub-op			
	da=sx+sy																0	0	0	Nop
	da=sx+sy+sa																0	0	1	Add
	da=sx+sy; sa=sx+sy;																0	1	0	AddSub
	da=(sx+sy)*sa																0	1	1	Mul
	da=-(sx+sy)*sa																1	0	0	MulN
	da=min(sx+sy,sa)																1	0	1	Min
	da=max(sx+sy,sa)																1	1	0	Max
Extremum	1	1	0	PS	x/n	SX				SY				N/S	SA	DA	Sub-op			
	da=ext(sx,sy)																0	0	0	Nop
	da=ext(sx,sy,sa)																0	0	1	Ext
	da=ext(sx,sa)*sy																0	1	0	Mul
	da=-ext(sx,sa)*sy																0	1	1	MulN
	da=ext(sx,sa)+sy																1	0	0	Add
	da=ext(sx,sa)-sy																1	0	1	Sub
	ext(sa,da) ?1=sx,tr=sy,lcs=lc																1	1	0	amax
type-match																				
1 1 0 PS 0 SX SY x x x 1 1 1																				
1 1 0 PS 0 x x x x x x x x x 1 1 1																				
1 1 0 PS 1 Type SY 0 ereg 1 1 1																				
1 1 1 PS x SX SY SA DA N/S Sub-op																				
Permute																				
Reserved																				

FIG. 6H

Control and specifier Extensions

	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mul	0	Pred	PL	Sxt	Syt	Rnd	Lt	S'	S'	S'	0	SA	DA	obs	0	0				

Add	0	Pred	PL	Sxt	Syt	Lt	Sub-ext		0	SA	DA	obs	0	0						
							+/-+/-+/- x													
							x V/S Rnd Fp													
							tr/ctl Gx Fp													

Nop (uadd)
 Mul/MulN
 Min/Max

Ext	0	Pred	PL	Sxt	Syt	tr-ctl	Gx	Sub-ext		0	SA	DA	obs	0	0					
								Lt	Fp											
								Rnd	V/S											

Add/sub
 Mul

	0	Pred	PL	Sxt	Syt	Pctl1		0	ereg	Pctl	0	0								
--	---	------	----	-----	-----	-------	--	---	------	------	---	---	--	--	--	--	--	--	--	--

Type/offset/permute extensions

	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	Pred	PL	x	Type: SX		Type: SY		0	SA	DA	x	0	1						
	0	Pred	PL	Psx	Permute: SX		Permute: SY		0	SA	DA	Psy	1	0						
	0	Pred	I/R I/R	prX	Offset: SX		Offset: SY		0	SA	DA	prY	1	1						

Type override
 permute override
 Offset override

Shadow DSP

	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	Op	PL	op	SA	ereg1		DA	ereg2		1	SA	DA	Sub-op						

nop

	1	1	0	PL	0	x	x	x	Rnd	x	x	x	x	0	SA	DA	1	1	1	
--	---	---	---	----	---	---	---	---	-----	---	---	---	---	---	----	----	---	---	---	--

FIG. 6I

Control instructions

	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
add,sub	L	Pred	0	0	0		RX					RY				RZ			+/-	0	
max,min	L	Pred	0	0	0		RX					RY				RZ			X/N	1	
Shift	L	Pred	0	0	1		RX					UI4				RZ			UI2	R/L	<Bit1,Bits9-6> ==UI5 (Shift Amount)
Logic	L	Pred	0	1	0		RX					RY				RZ			&	&1	
Mux	L	Pred	0	1	1		RX					RY				RZ			Pd	0	
mov	L	Pred	0	1	1		RX					DZ			Rxt	Dzl	0	0	0	1	
addi	L	Pred	0	1	1		SI4					DZ			x	x	1	0	0	1	
mov2erg	L	Pred	0	1	1		RX				unit	ereq			qd	type		1	0	1	
l.dm	L	Pred	0	1	1		RX					DZ1				DZ2			1	1	
Set4bits	L	Pred	1	0	0		UI4:POS					RZ			Rzt			UI4		0	
Set2bits	L	Pred	1	0	0		UI4:POS					RZ			Rzt	UI2		0	0	1	
Setbit	L	Pred	1	0	0		UI4:POS					RZ			Rzt	UI1	UI1	1	0	1	<Bit3,Bits13-10>==UI5 POS
Movi	L	Pred	1	0	0					SI8						RZ			1	1	
Jmp	L	Pred	1	0	1					SI9						0		PRED	0	0	
Call	L	Pred	1	0	1					SI9						1		PRED	0	0	
Loop	L	Pred	1	0	1		UI5:Lcount					UI5:Lsize						UI2:Lst	0	1	
Jmpi	L	Pred	1	0	1		RX				x	x	x	x	x	0		PRED	1	0	
Calli	L	Pred	1	0	1		RX				x	x	x	x	x	1		PRED	1	0	
Loopi	L	Pred	1	0	1		RX				x		UI5:Lsize					UI2:Lst	1	1	
Test	L	Pred	1	1	0		RX					RY				PZ		=,<,>		0	
Testbit	L	Pred	1	1	0		RX					UI5				PZ		B	0	1	
Andp, orp	L	Pred	1	1	0		Pa			Pb		Pc				PZ		&	1	1	
Load	L	Pred	1	1	1		MX					RZ				Ext		0	0	0	
Store	L	Pred	1	1	1		MZ					RZ				Ext		1	0	0	
eLoad	L	Pred	1	1	1		MX					RY			1	1	1	0	0	0	
eStore	L	Pred	1	1	1		MZ					RY			1	1	1	1	0	0	
Extended	L	Pred	1	1	1							Bits 27:16							1	0	
Logic2	L	Pred	1	1	1		RX					RY/RZ			Rxt	Ryt	&1,&1		0	1	
mov-erg	L	Pred	1	1	1		unit	ereq				RZ			qd	Sft		0	1	1	
Crb	L	Pred	1	1	1		RX					RZ			s/m	0	0	1	1	1	
Panty	L	Pred	1	1	1		RX					PZ	0/E	0	1	0	1	1	1	1	
Stm	L	Pred	1	1	1		MZ					RX			1	1	0	1	1	1	
Abs	L	Pred	1	1	1		RX					RZ			0	0	1	1	1	1	
Neg	L	Pred	1	1	1		RX					RZ			0	1	1	1	1	1	
Div-step	L	Pred	1	1	1		RX					RZ			1	0	1	1	1	1	
Test&Set	L	Pred	1	1	1		RX					PZ		0	1	1	1	1	1	1	
Reserved	L	Pred	1	1	1							0	0	1	1	1	1	1	1	1	
Return	L	Pred	1	1	1		Pred	I-ctl	0	1	0	1	1	1	1	1	1	1	1	1	
Zero-ac	L	Pred	1	1	1		ac #				1	1	0	1	1	1	1	1	1	1	
eSync	L	Pred	1	1	1		RZ				0	1	1	1	1	1	1	1	1	1	
Swi	L	Pred	1	1	1		UI3	0	1	1	1	1	1	1	1	1	1	1	1	1	
Nop	L	Pred	1	1	1		UI3	1	1	1	1	1	1	1	1	1	1	1	1	1	

FIG. 6J

Extended Control

Bits 13:2 of upper half (39:20)

13	12	11	10	9	8	7	6	5	4	3	2	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX				RZ				0	0	0	0	0	x	x	0	Rxt	Rzt	I/E	R/I	R/I	Offset:U15				Length:U15				0		

Insert/Extract

Inserti
Shift

UI4 length								0	0	0	1	0	x	x	0	Rzt																
RX								0	0	0	0	0	rxh	rxl	0	rxh	rxl	0	U/S	1												

Rotate

RX								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
----	--	--	--	--	--	--	--	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

jmp, call
 loop
 loopi
 mult
 add/sub
 Reserved
 logicp
 Testi
 Movi
 loadi
 storei
 loadt
 storet
 Addi/subi
 mini,maxi
 andi,on

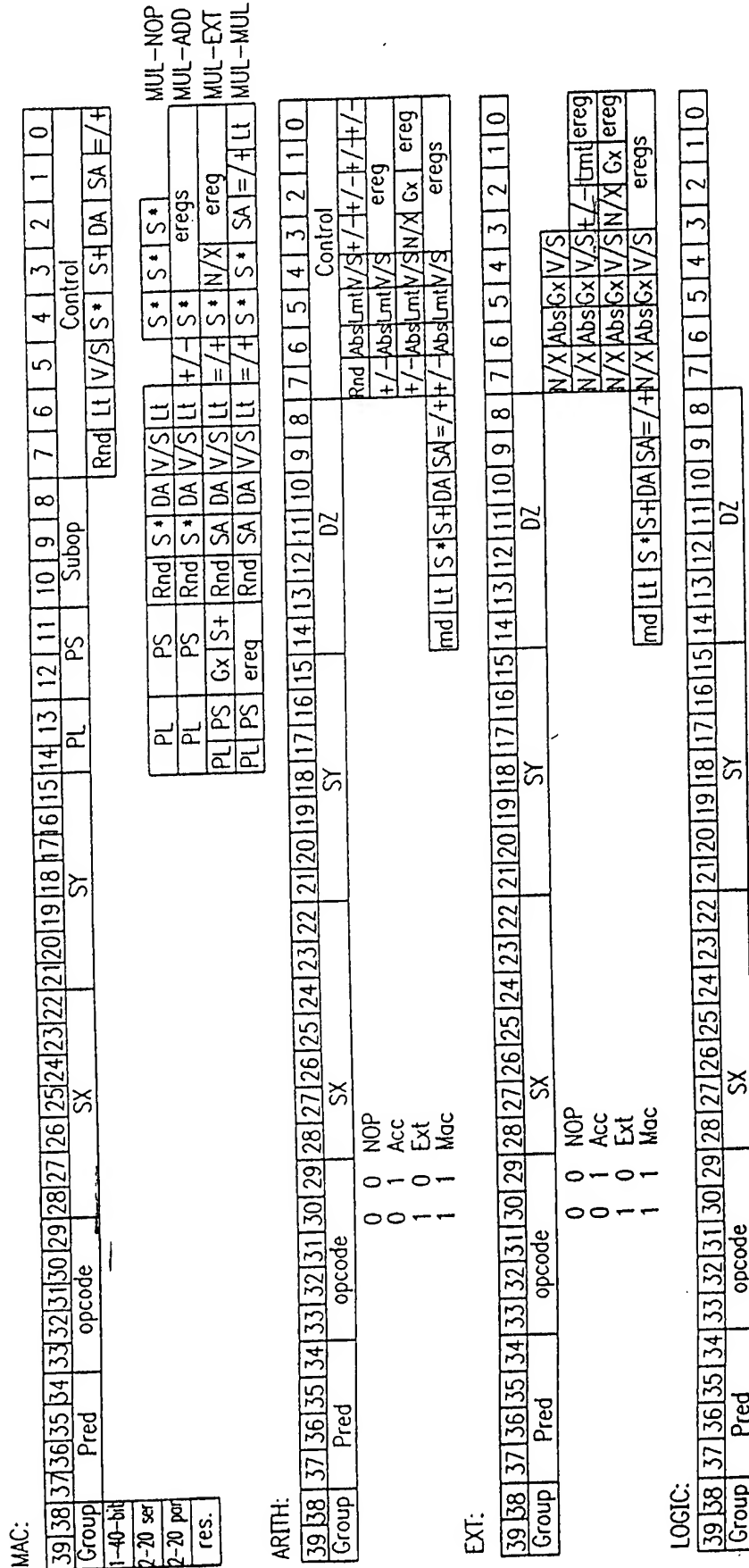
UI4 outer LC	UI4 inner LC	J/C	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX	RY		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX	RY		0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX	RY		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX	D	PZ	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX	D	PZ	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H/L	Fill	RZ	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type		RZ	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type		RZ	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX		RZ	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M7		RZ	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX		RZ	1	0	+	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX		RZ	1	0	X/N	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RX		RZ	1	1	&i	H/L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fill: Sign/
Zero

Bit 15 is
continuation
of inner LC

andp,orp,
andorp,
orandp,
pz=(px
relop py)
relop pv)

FIG. 6K



[illegible]

Immediate:

Immediate:																																							
39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Group		op	DZ		Imm32																																		
Group		opcode	SX		DZ		Subop										Imm16																						

Test:

Test:		UK																																					
Group	Pred	33	32	31	30	29	28	27	26	25	24	23	22	SY				DPz				Subop																	
39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

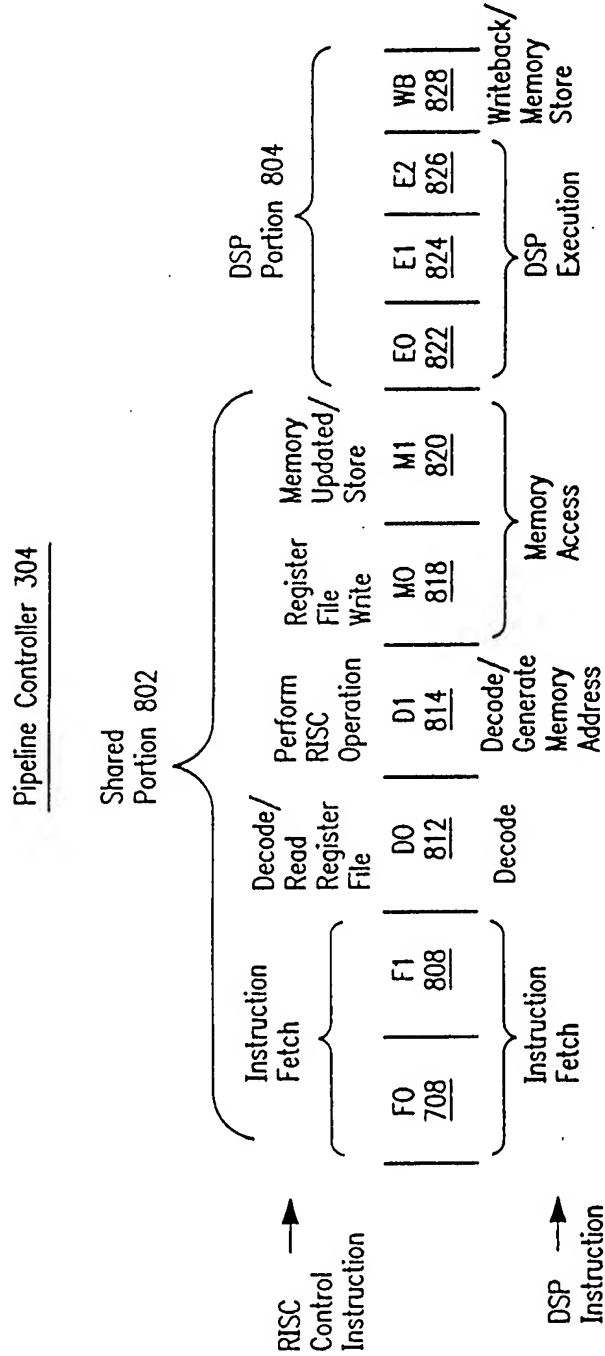
Branch:

[illegible]

Misc:

FIG. 6L-2





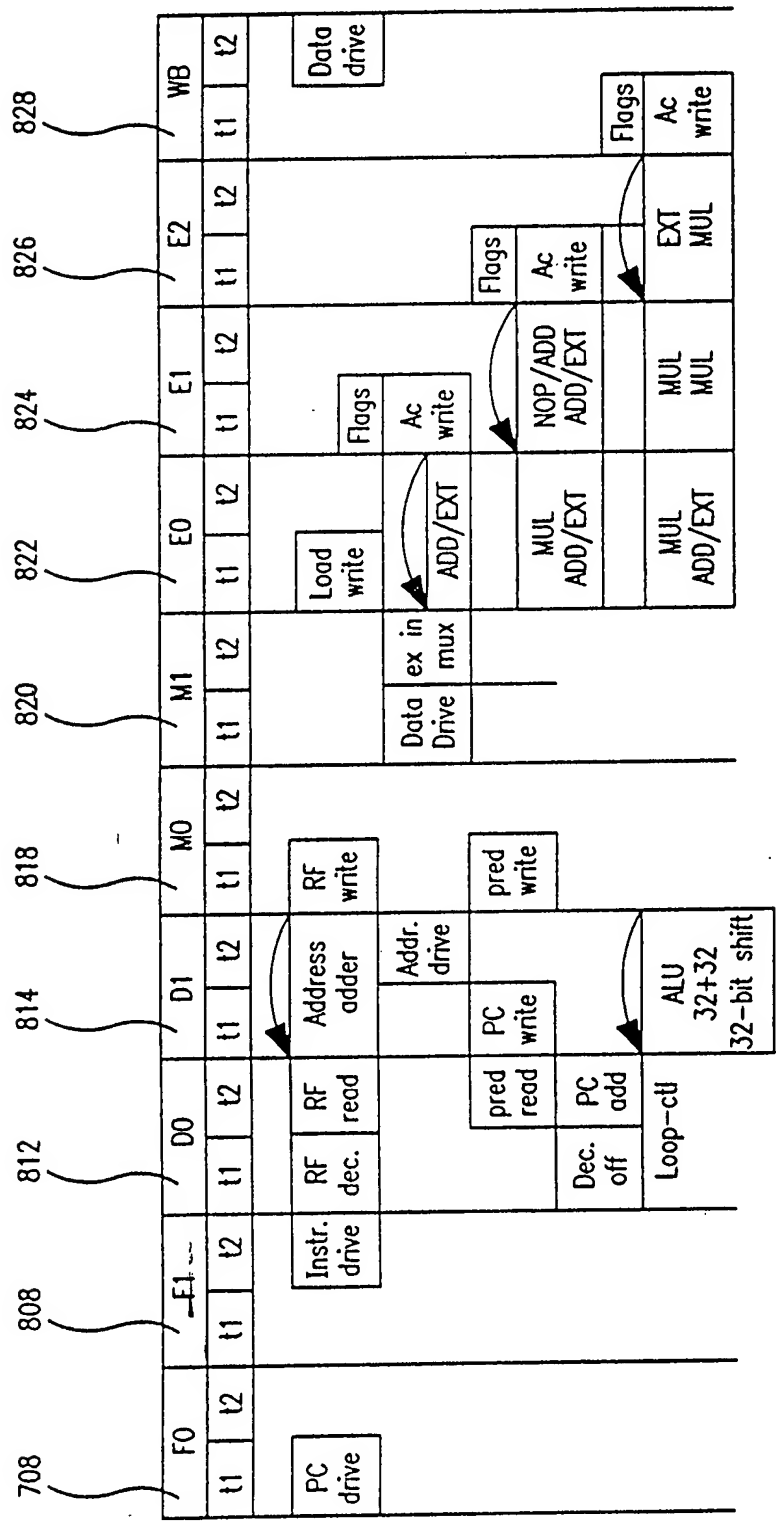


FIG. 8b

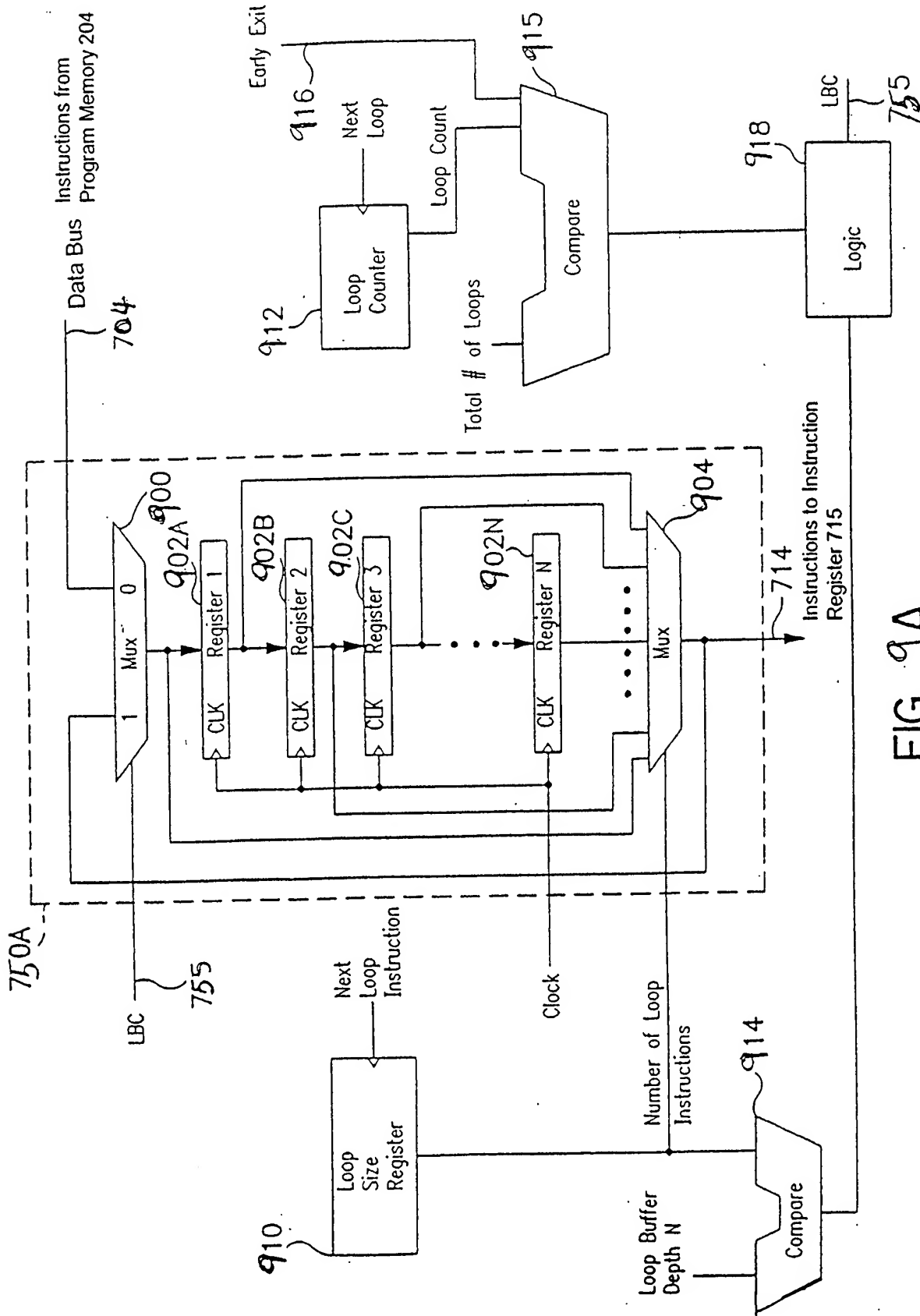


FIG. 9A

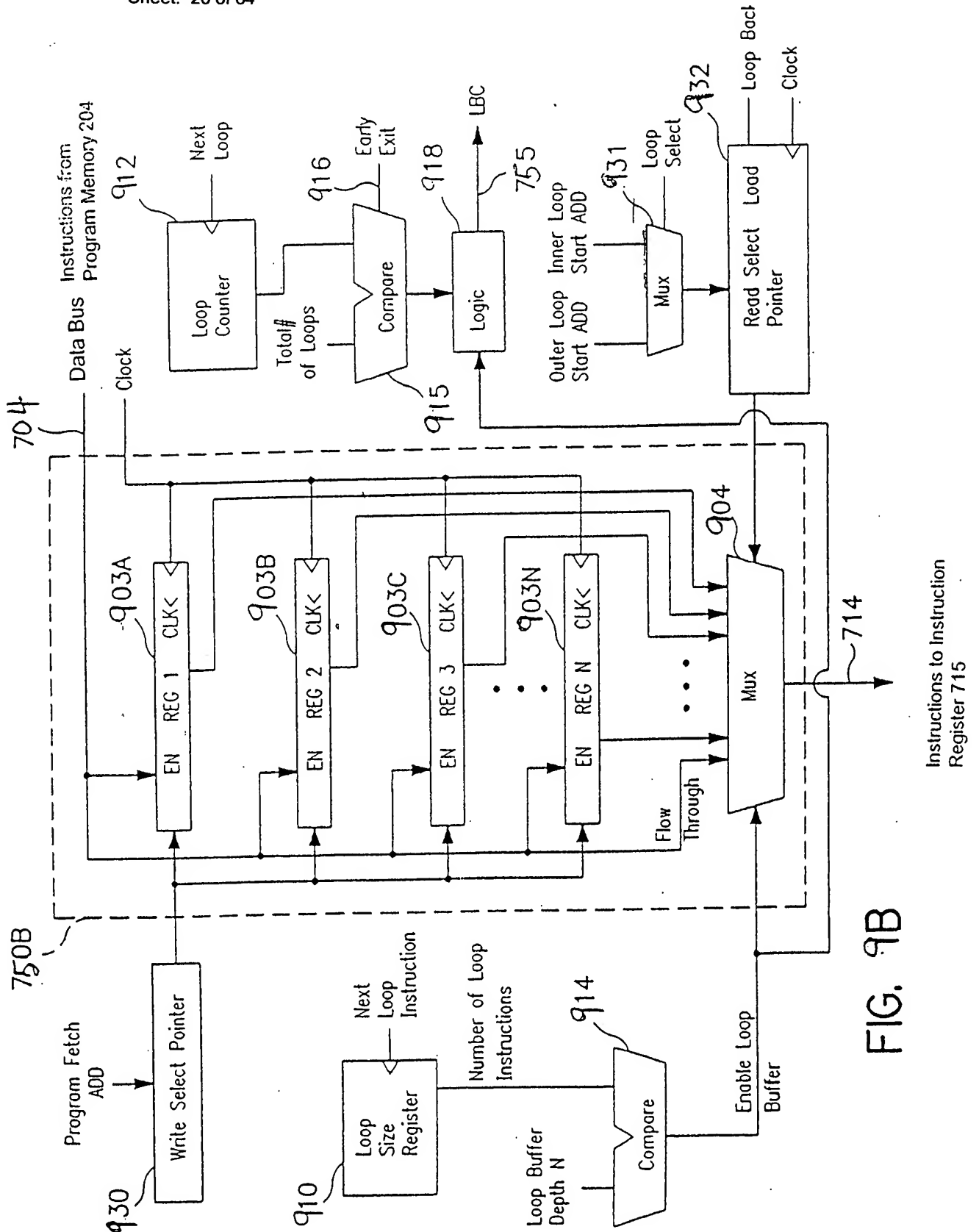


FIG. 9B

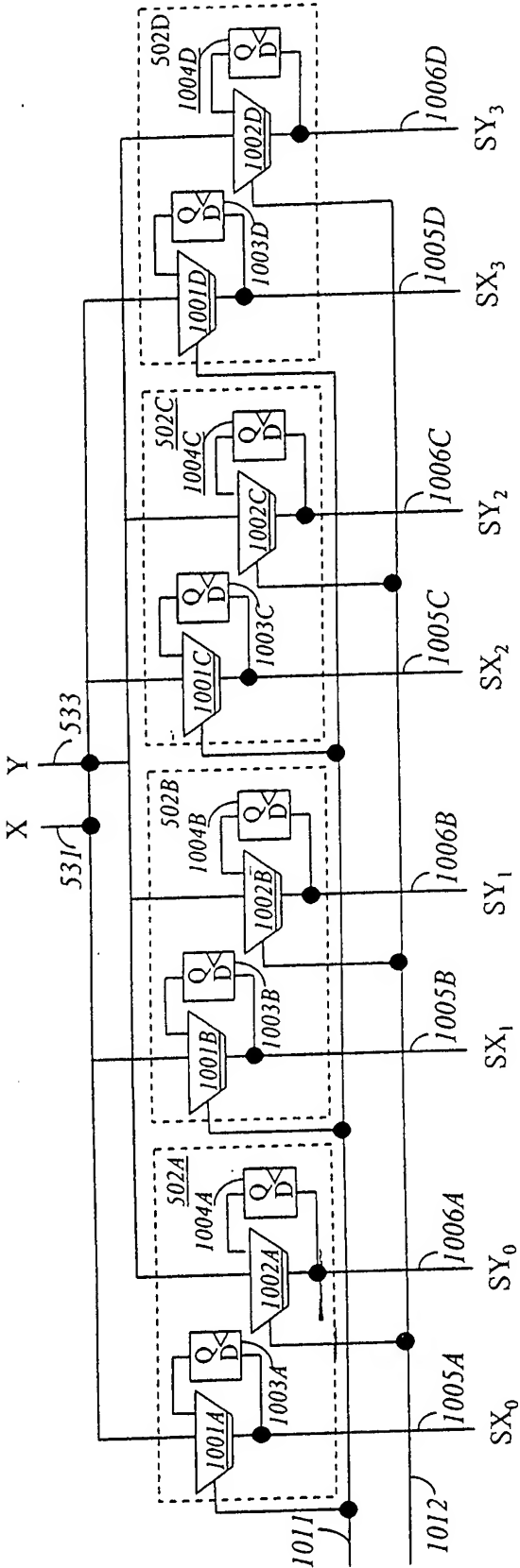


FIG. 10

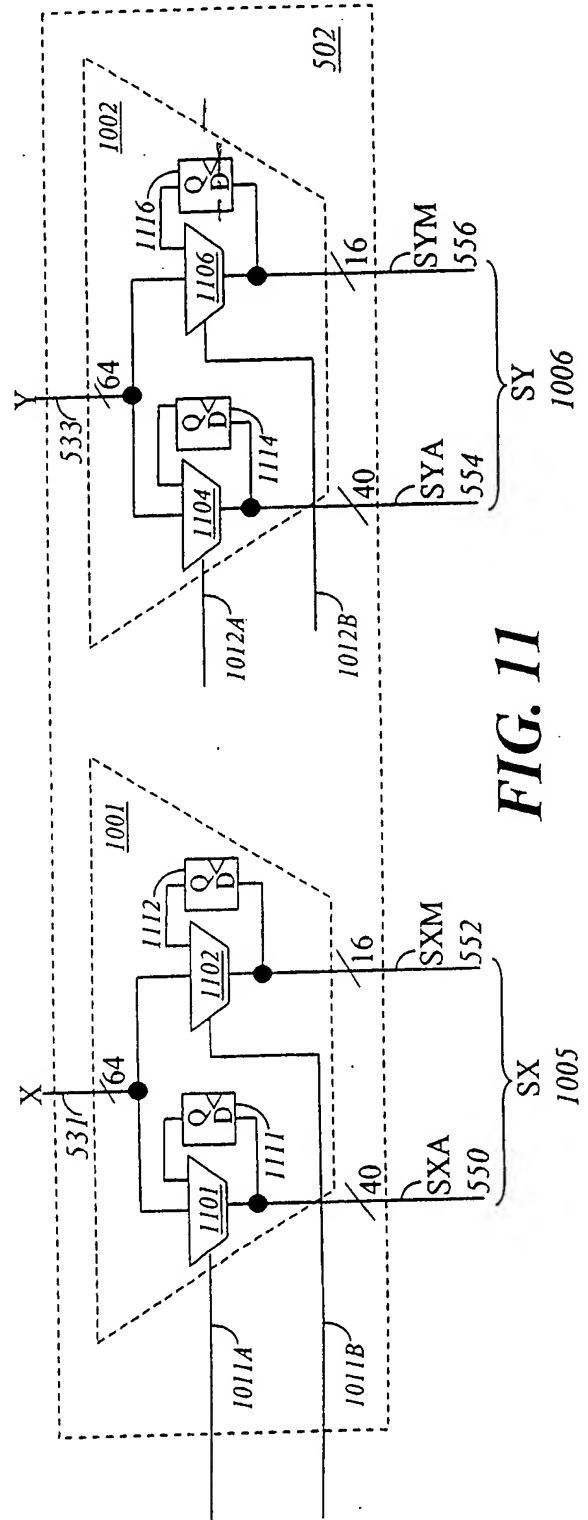
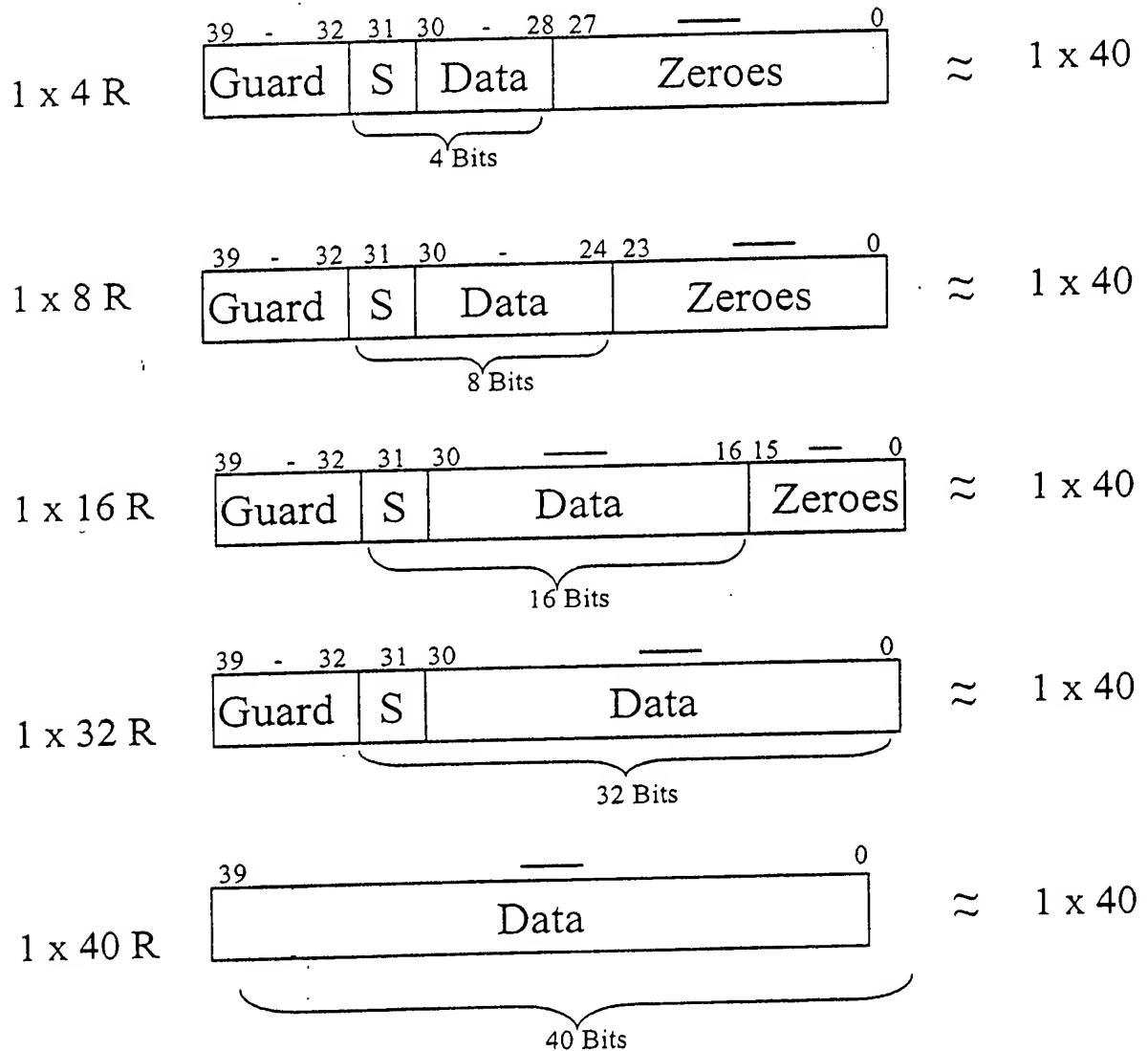


FIG. 11

FIG. 12A

Data Type

SP Configuration



SXA 550 or SYA 554

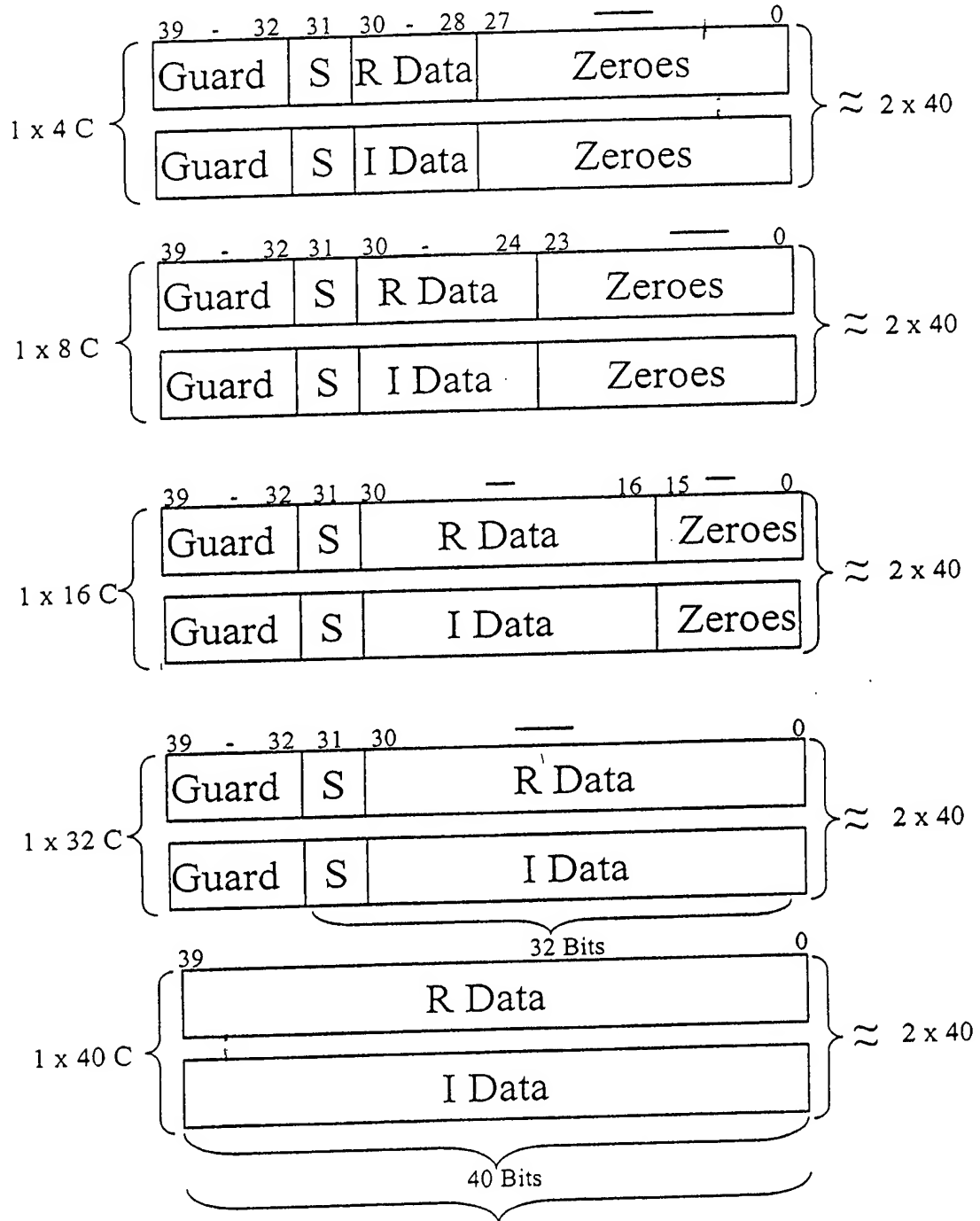
FIG. 12B

<u>Data Type</u>		<u>SP Configuration</u>
1 x 4 R		$\approx 1 \times 16$
1 x 8 R		$\approx 1 \times 16$
1 x 16 R		$\approx 1 \times 16$
1 x 32 R		$\approx 1 \times 16$
1 x 40 R		$\approx 1 \times 16$
	SXM 552A – 552B or SYM 556A – 556B	

FIG. 12C

Data Type

SP Configuration

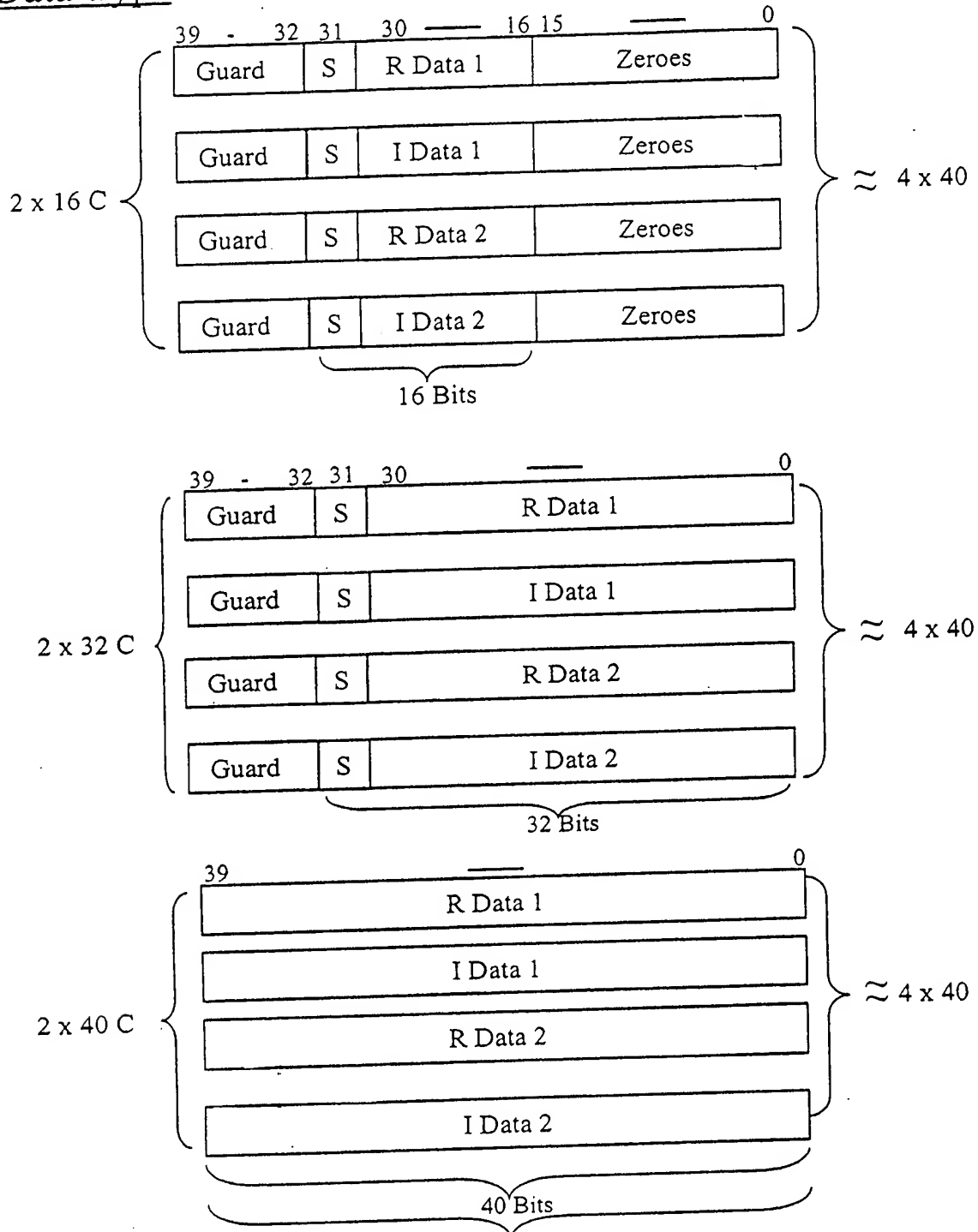


SXA 550A and SXA 550B
 or
 SYA 554A and SYA 554B

FIG. 12D

Data Type

SP Configuration



SXA 550A, SXA 550B, SXA 550C, and SXA 550D
 or
 SYA 554A, SYA 554B, SYA 554C, and SYA 554D

FIG. 12E

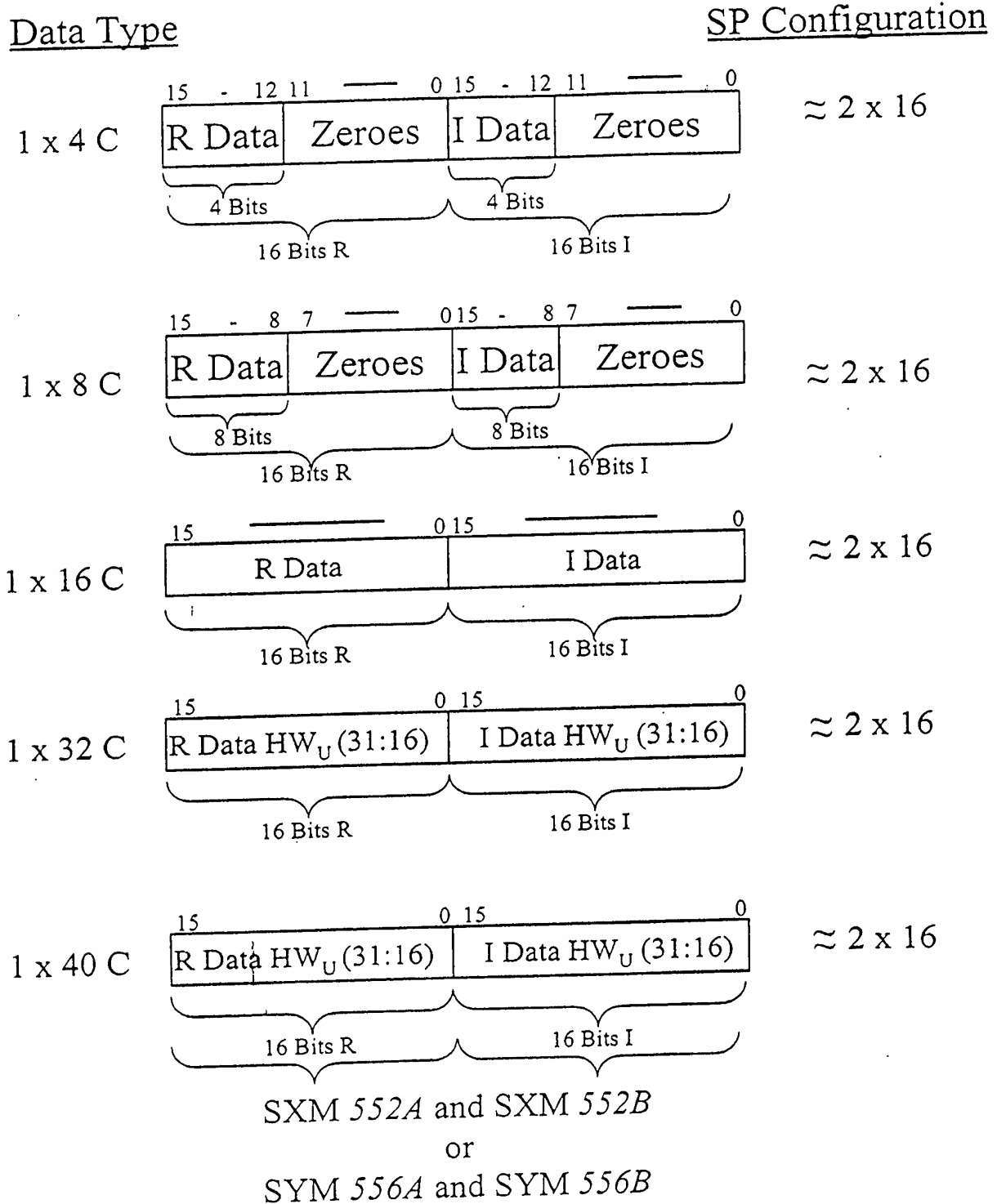
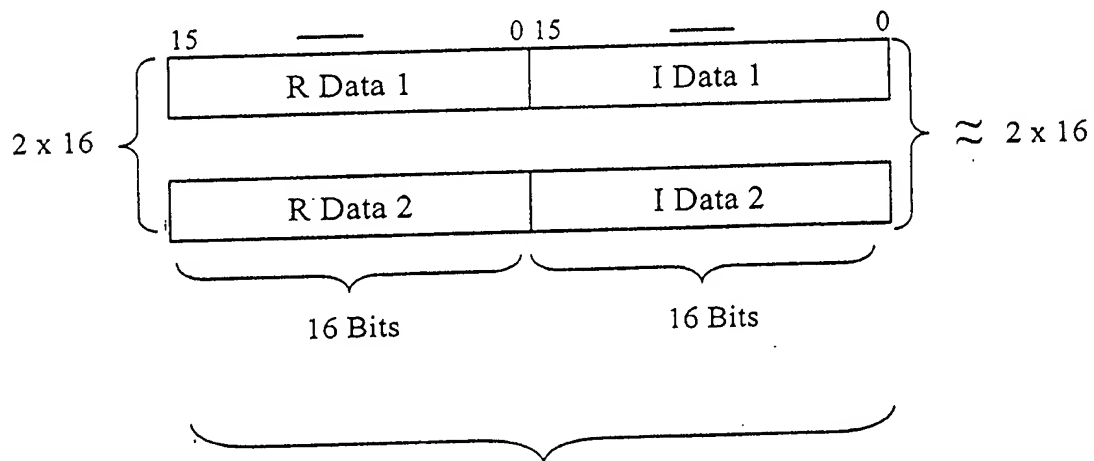
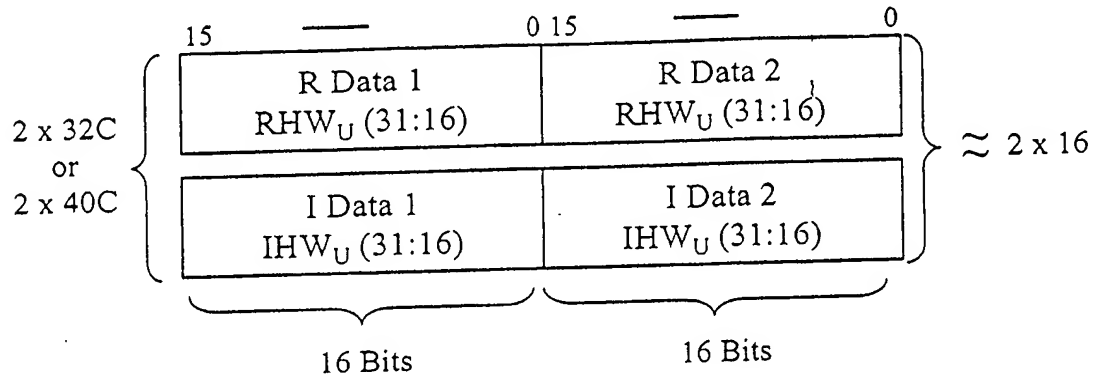


FIG. 12F

Data Type

SP Configuration



SXM 552A, SXM 552B, SXM 552C, and SXM 552D
 or
 SYM 556A, SYM 556B, SYM 556C, and SYM 556D

Operand 1 Data Type: $N_1 \times S_1 R$
Operand 2 Data Type: $N_2 \times S_2 R$
Type Matching R: $\text{Max}(N_1 \text{ or } N_2) \times \text{Max}(S_1 \text{ or } S_2) R$

Fig. 13A

Operand 1 Data Type: $N_1 \times S_1 C$
Operand 2 Data Type: $N_2 \times S_2 C$
Type Matching C: $\text{Max}(N_1 \text{ or } N_2) \times \text{Max}(S_1 \text{ or } S_2) C$

Fig. 13B

Operand 1 Data Type: $N_1 \times S_1 R$
Operand 2 Data Type: $N_2 \times S_2 C$
Type Matching R+C: $\text{Max}(N_1 \text{ or } N_2) \times \text{Max}(S_1 \text{ or } S_2) C$

Fig. 13C

	1x16 real	2x16 real	1x16 cmpx	4x16 real	2x16 cmpx	1x32 real	2x32 real	1x32 cmpx	4x32 real	2x32 cmpx	1x40 real	2x40 real	1x40 real	4x40 real	2x40 cmpx
1x16 real	1 unit	2 unit	2 unit	4 unit	4 unit	2 unit	4 unit	4 unit							
2x16 real	2 unit	2 unit													
1x16 cmpx	2 unit		4 unit												
4x16 real	4 unit			4 unit											
2x16 cmpx	4 unit														
1x32 real	2 unit														
2x32 real	4 unit														
1x32 cmpx	4 unit														
4x32 real															
2x32 cmpx															
1x40 real															
2x40 real															
1x40 real															
4x40 real															
2x40 real															

FIG. 14

	1x16 real	2x16 real	1x16 cmpx	4x16 real	2x16 cmpx	1x32 real	2x32 real	1x32 cmpx	4x32 real	2x32 cmpx	1x40 real	2x40 real	4x40 real	2x40 cmpx
1x16 real	1 unit	2 unit		4 unit		1 unit	4 unit		4 unit		1 unit	2 unit	4 unit	
2x16 real	2 unit	2 unit				2 unit	2 unit					2 unit		
1x16 cmpx														
4x16 real	4 unit			4 unit		4 unit			4 unit				4 unit	
2x16 cmpx														
1x32 real	1 unit	2 unit		4 unit		1 unit	2 unit		4 unit		1 unit	2 unit	4 unit	
2x32 real	4 unit	2 unit				2 unit	2 unit					2 unit		
1x32 cmpx														
4x32 real	4 unit			4 unit		4 unit			4 unit		4 unit		4 unit	
2x32 cmpx														
1x40 real	1 unit					1 unit			4 unit		1 unit			
2x40 real	2 unit	2 unit				2 unit	2 unit					2 unit		
1x40 real														
4x40 real	4 unit					4 unit			4 unit				4 unit	
2x40 real														

FIG. 15A

	1x16 real	2x16 real	1x16 cmpx	4x16 real	2x16 cmpx	1x32 real	2x32 real	1x32 cmpx	4x32 real	2x32 cmpx	1x40 real	2x40 real	1x40 real	4x40 real	2x40 cmpx
1x16 real	1 unit	2 unit	2 unit	4 unit	4 unit	1 unit	2 unit	2 unit	4 unit	4 unit	1 unit	2 unit	2 unit	4 unit	4 unit
2x16 real	2 unit	2 unit				2 unit						2 unit			
1x16 cmpx	2 unit		2 unit								2 unit		2 unit		
4x16 real	4 unit			1 unit		4 unit			4 unit					4 unit	
2x16 cmpx	4 unit				4 unit					4 unit					4 unit
1x32 real	1 unit	2 unit				1 unit	2 unit	2 unit	4 unit		1 unit	2 unit	2 unit	4 unit	
2x32 real	2 unit	2 unit				2 unit	2 unit					2 unit			
1x32 cmpx	2 unit		2 unit			2 unit					2 unit		2 unit		
4x32 real	4 unit			4 unit		4 unit			4 unit		4 unit			4 unit	
2x32 cmpx	4 unit				4 unit					4 unit					4 unit
1x40 real	1 unit					1 unit					1 unit	2 unit		4 unit	
2x40 real	2 unit	2 unit				2 unit	2 unit				2 unit	2 unit			
1x40 real	2 unit		2 unit			2 unit							2 unit		
4x40 real	4 unit			4 unit		4 unit			4 unit		4 unit			4 unit	
2x40 real	4 unit				4 unit					4 unit					4 unit

FIG. 15B

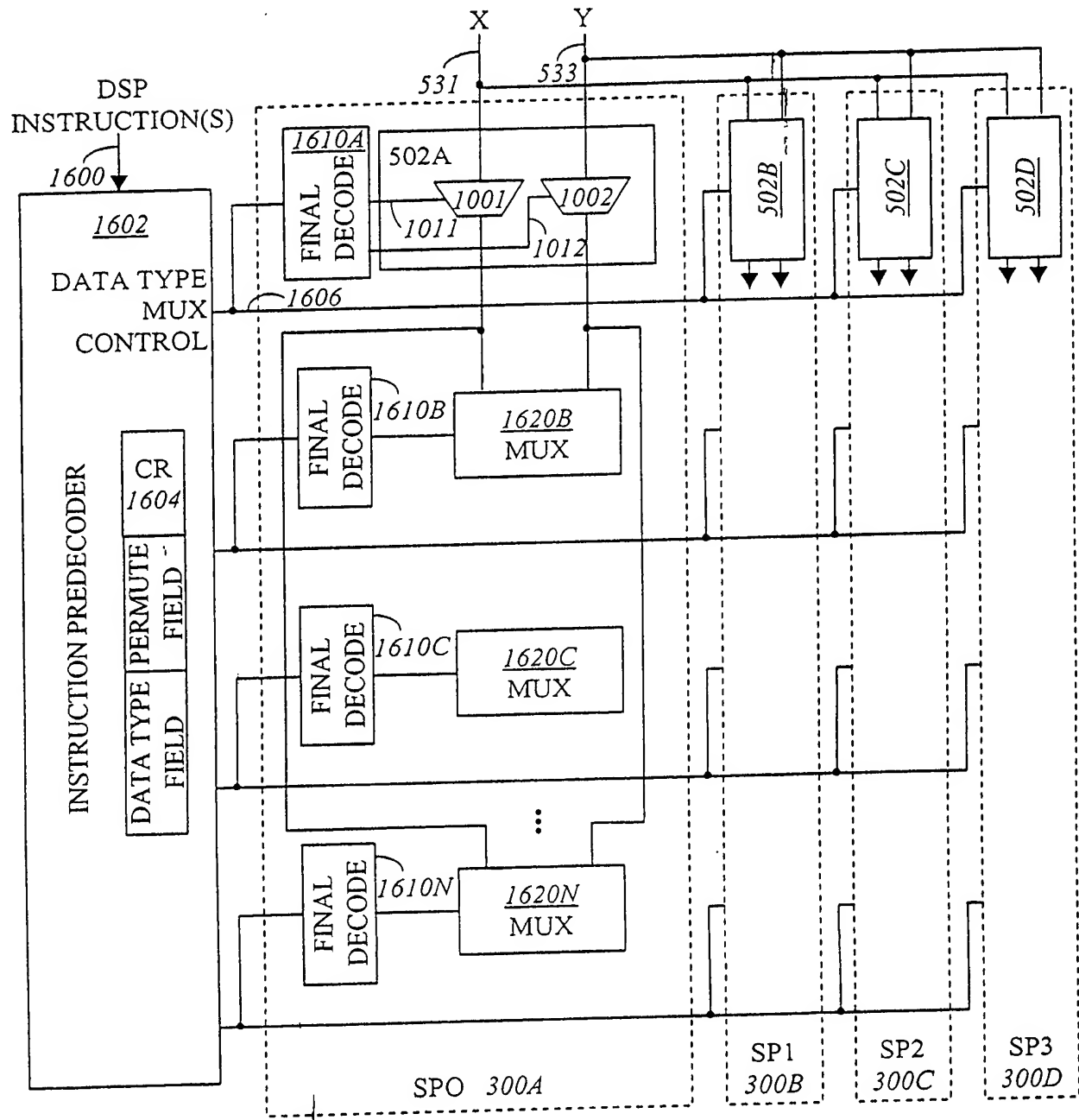


FIG. 16

Data Type: N x S(R/C)

FIG. 17

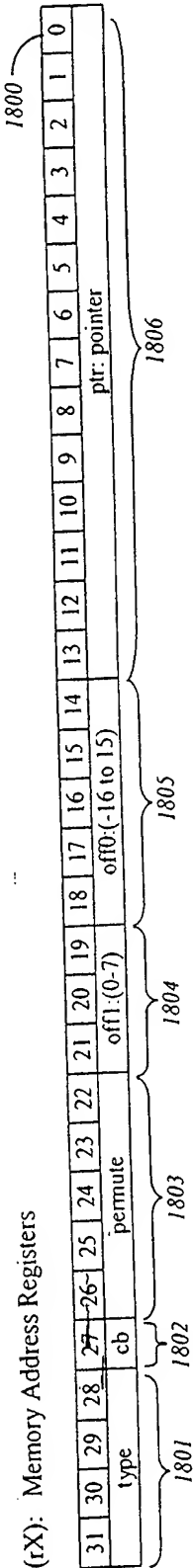


FIG. 18

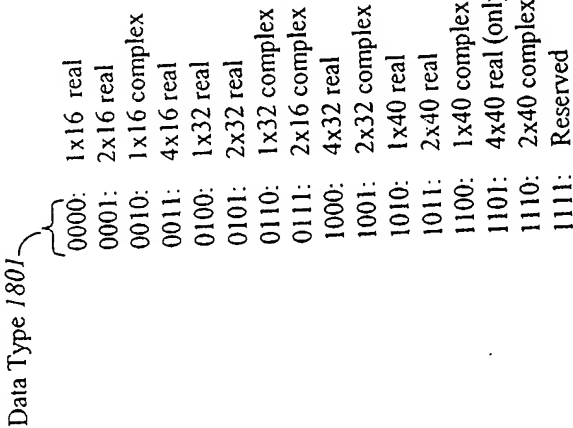


FIG. 19

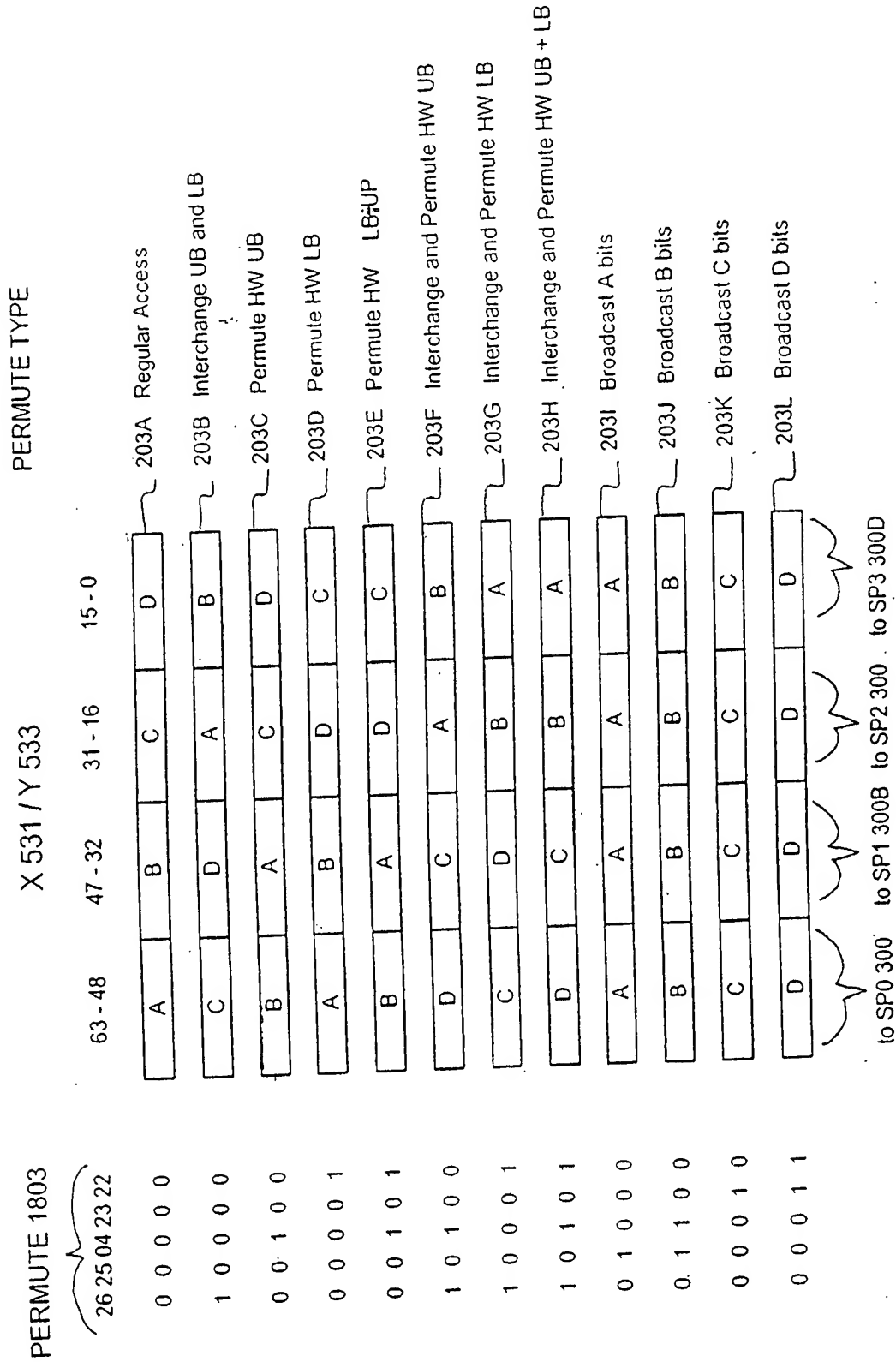


FIG. 20

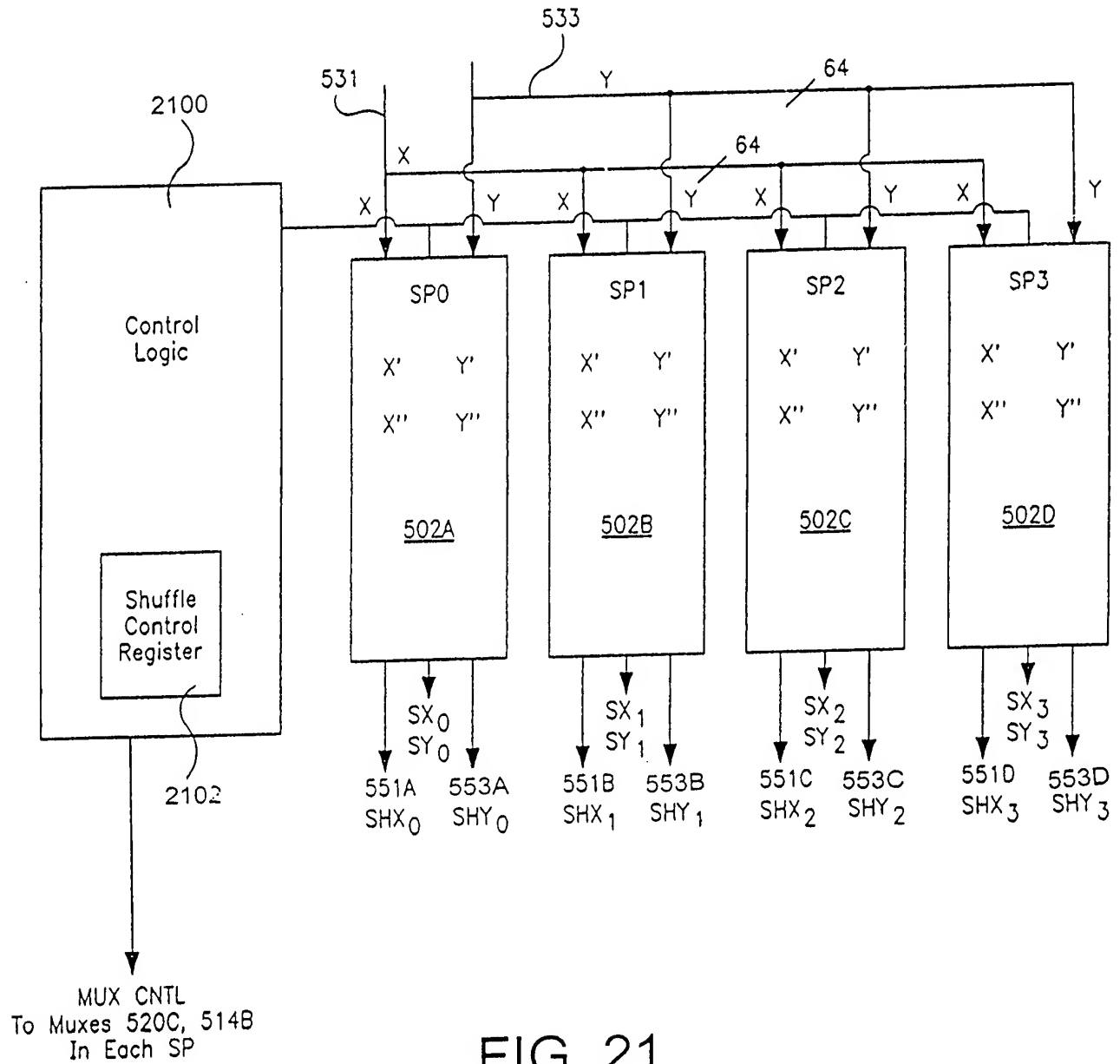


FIG. 21

$$X'=[SX_{10}, SX_{11}, SX_{12}, SX_{13}] \text{e.g. } [X_0, X_1, X_2, X_3]$$
$$X''=[SX_{20}, SX_{21}, SX_{22}, SX_{23}] \text{e.g. } [X_4, X_5, X_6, X_7]$$

Where SX_{ab} :S=Source; a=delay; b= SP unit number(e.g. SP3,SP2,SP1,SP0; or termed U3,U2,U1,U0)

$$Y'=[SY_{10}, SY_{11}, SY_{12}, SY_{13}]$$
$$Y''=[SY_{20}, SY_{21}, SY_{22}, SY_{23}]$$

Where SY_{ab} :S=Source; a=delay; b= SP unit number(e.g. SP3,SP2,SP1,SP0; or termed U3,U1,U0)

FIG. 22A

$$\text{FIR Filter} \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_N \end{bmatrix} * \begin{bmatrix} Y_0 \\ \vdots \\ Y_N \end{bmatrix} = X_0 Y_0 + X_1 Y_1 + \dots + X_N Y_N$$

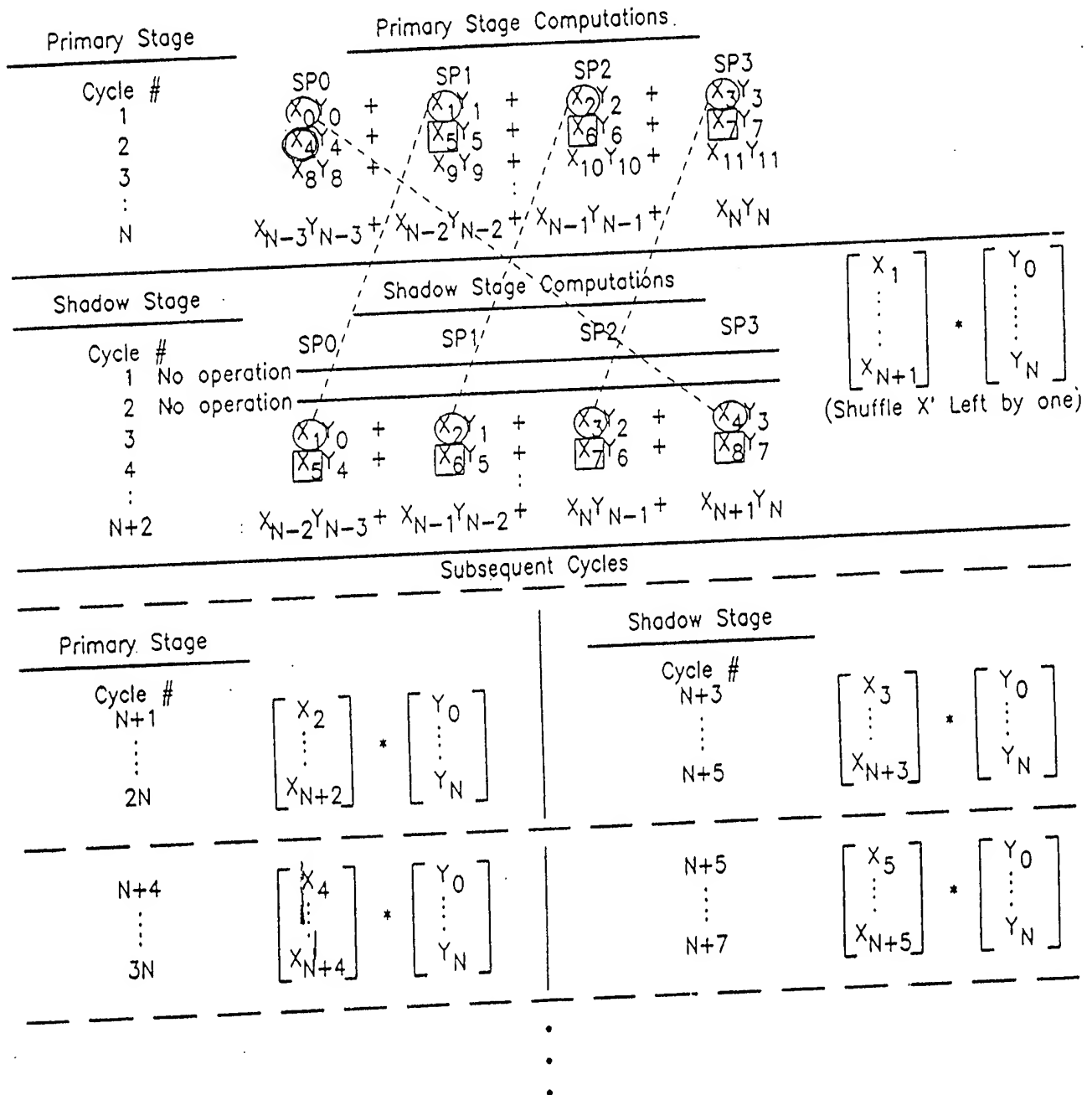


FIG. 22B

shuffle		Shuffle Control Register																															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
u3		u2		u1		u0		u3		u2		u1		u0		u3		u2		u1		u0		u3		u2		u1		u0			
SY2S								SY1S								SX2S								SX1S									

Units are connected to their nearest neighbors for shuffling the sources using the following bit diagram:

- 00
- Unit N+1, SX1=X' (right)
- 01
- Unit N+1, SX2=X'' (right)
- 10
- Unit N-1, SX1=X' (left)
- 11
- Unit N-1, SX2=X'' (left)

For example to shift the sources to the left by one:

3	2	1	0	From
2	1	0	3	Into

The bits should be 10101010 (\$AA)

FIG. 22C

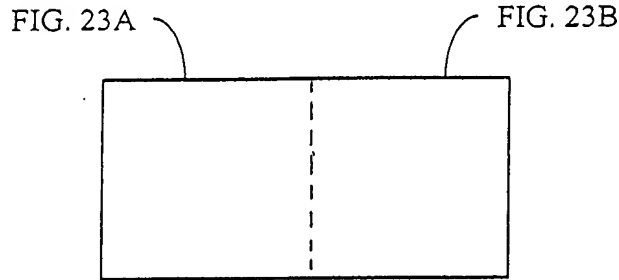


FIG. 23

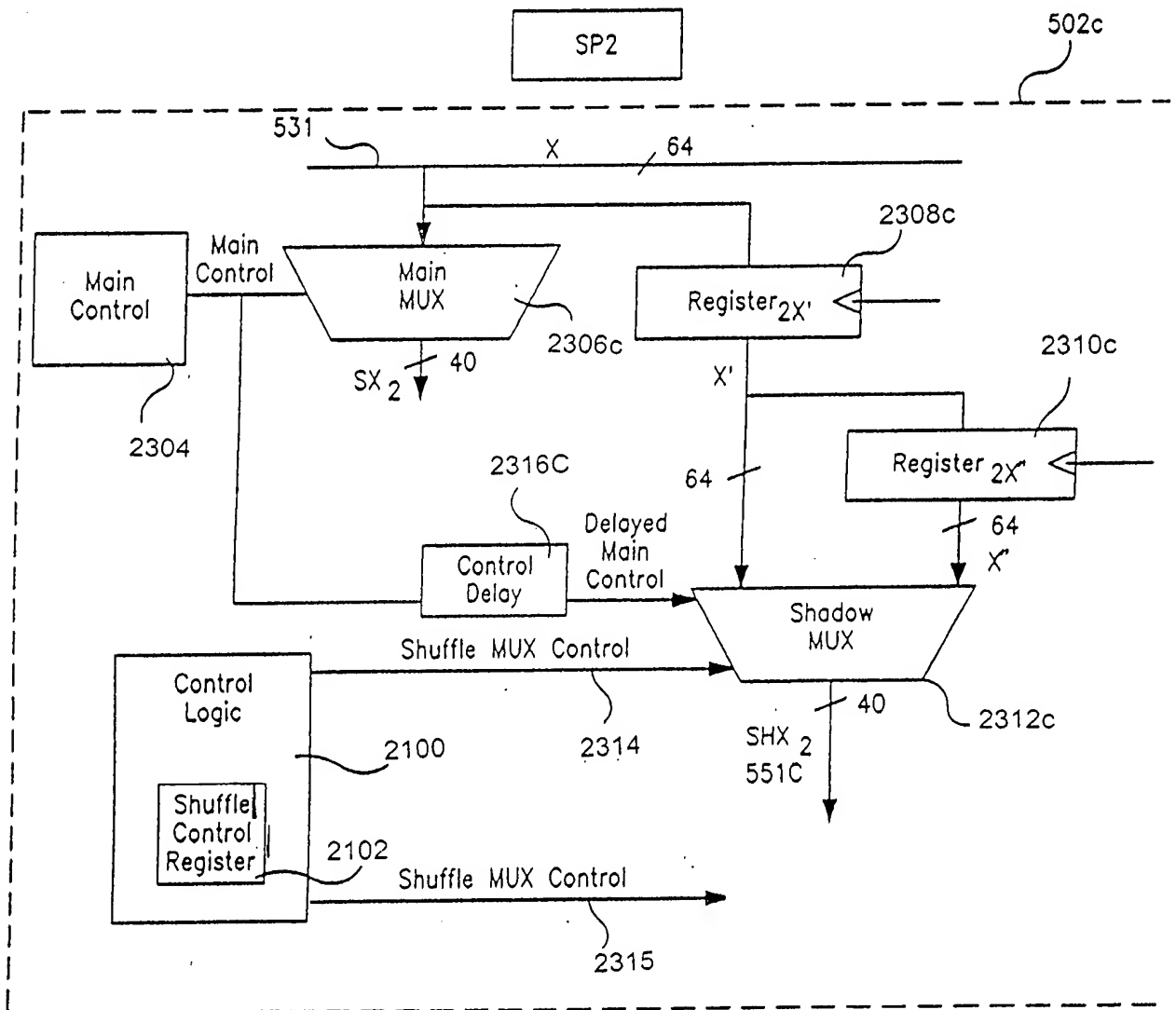


FIG. 23A

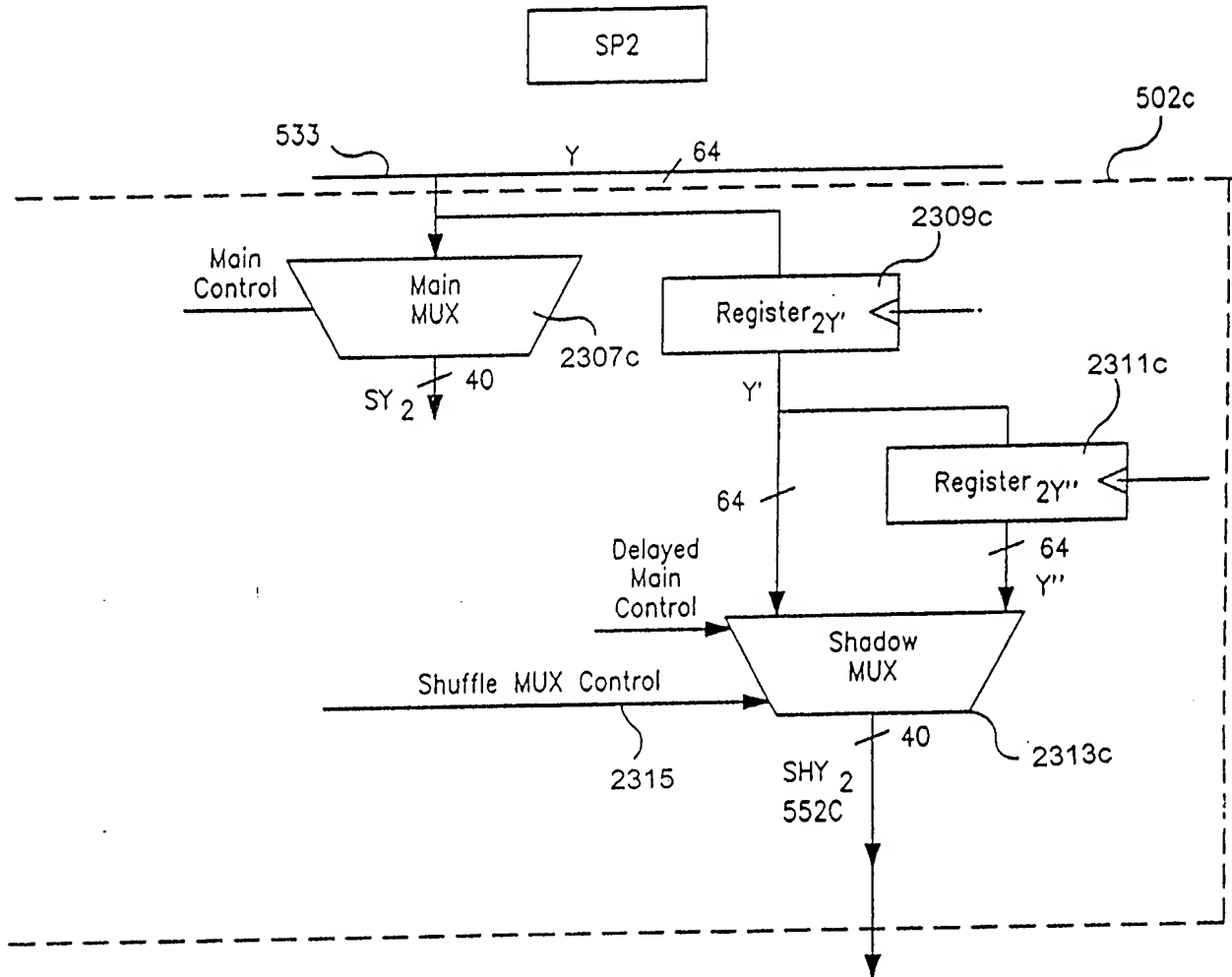
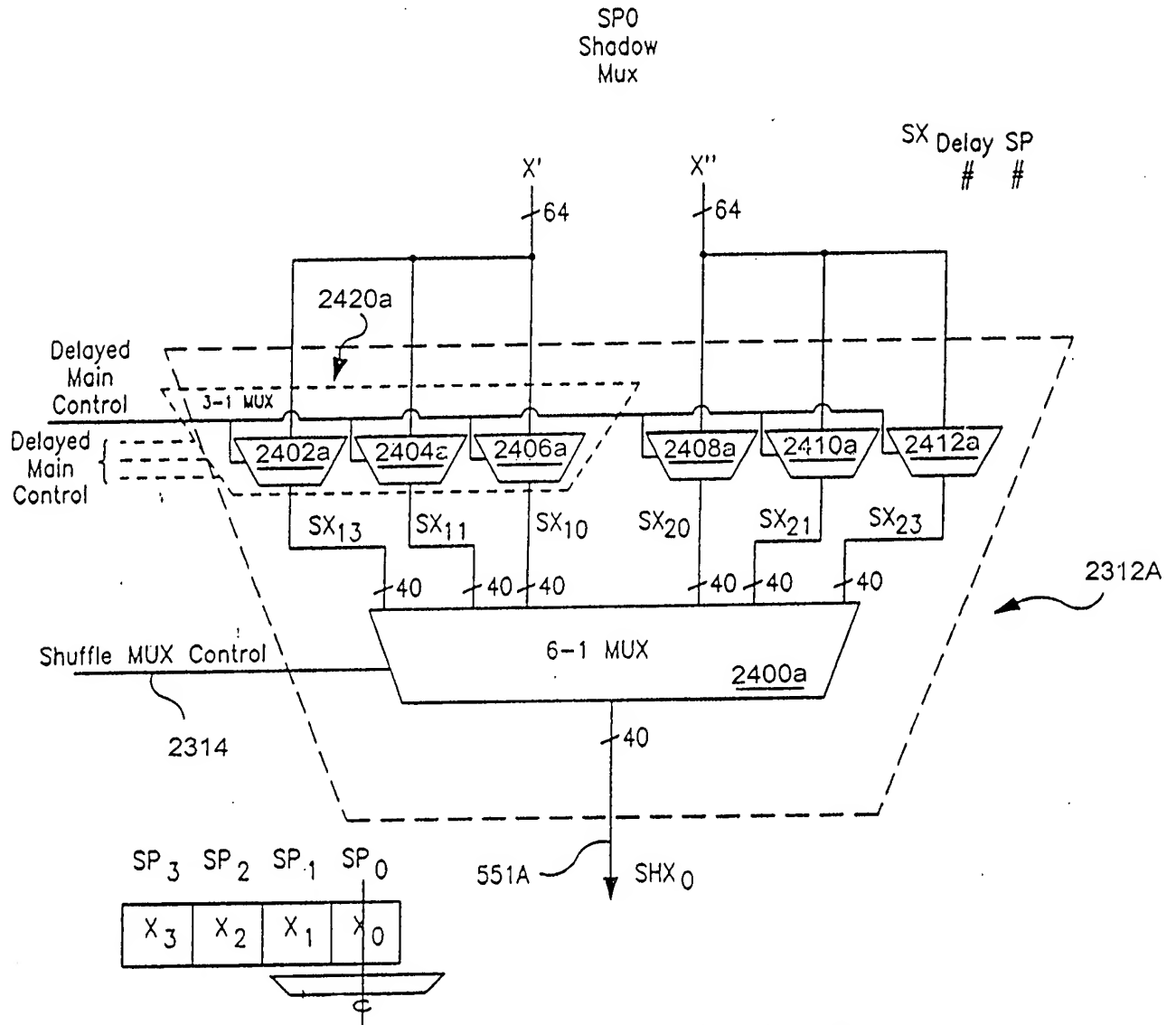


FIG. 23B



$$\begin{aligned} X_0 &= SX_{10}, SX_{20} \\ X_1 &= SX_{11}, SX_{21} \\ X_3 &= SX_{13}, SX_{23} \end{aligned}$$

FIG. 24A

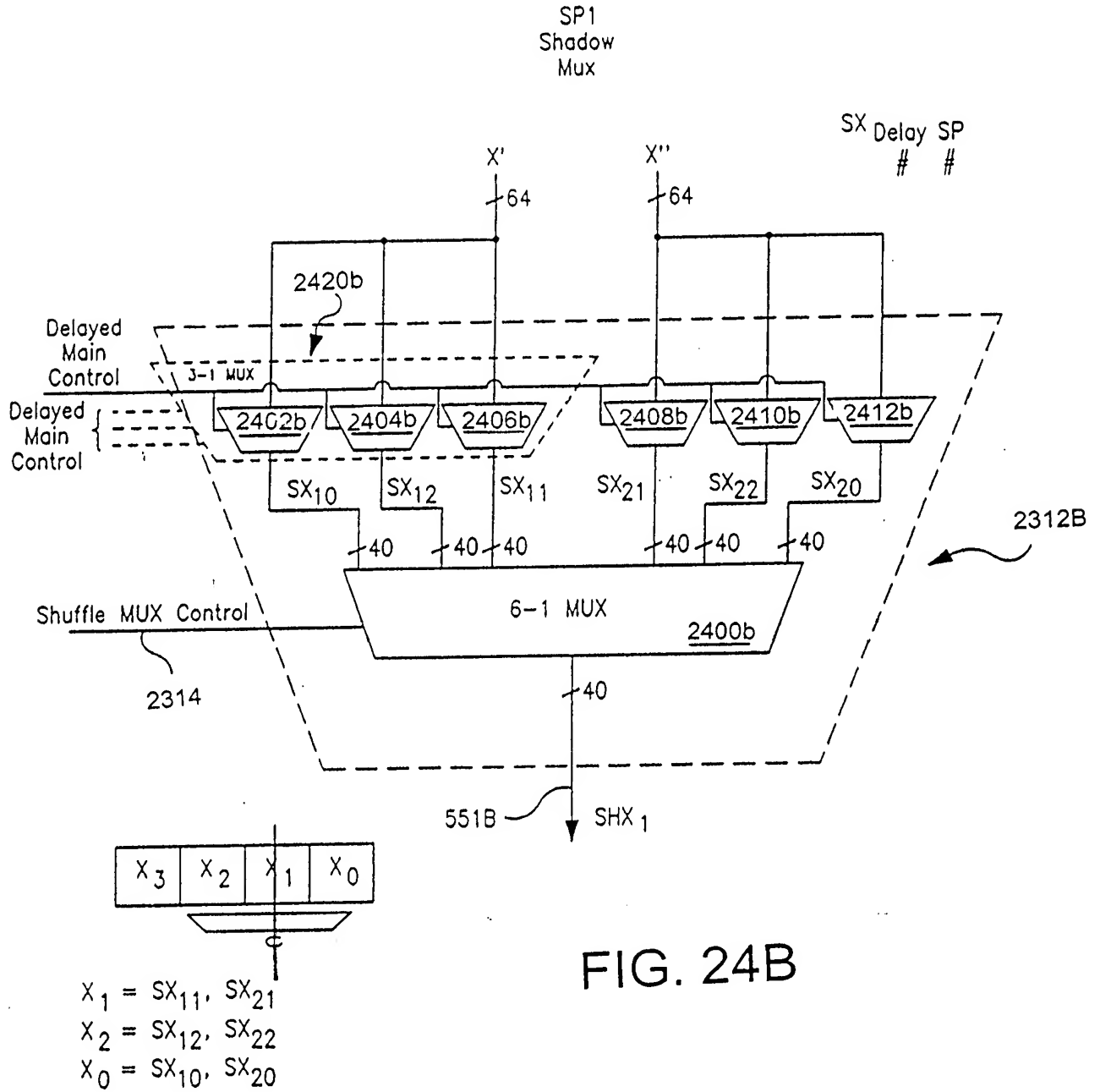
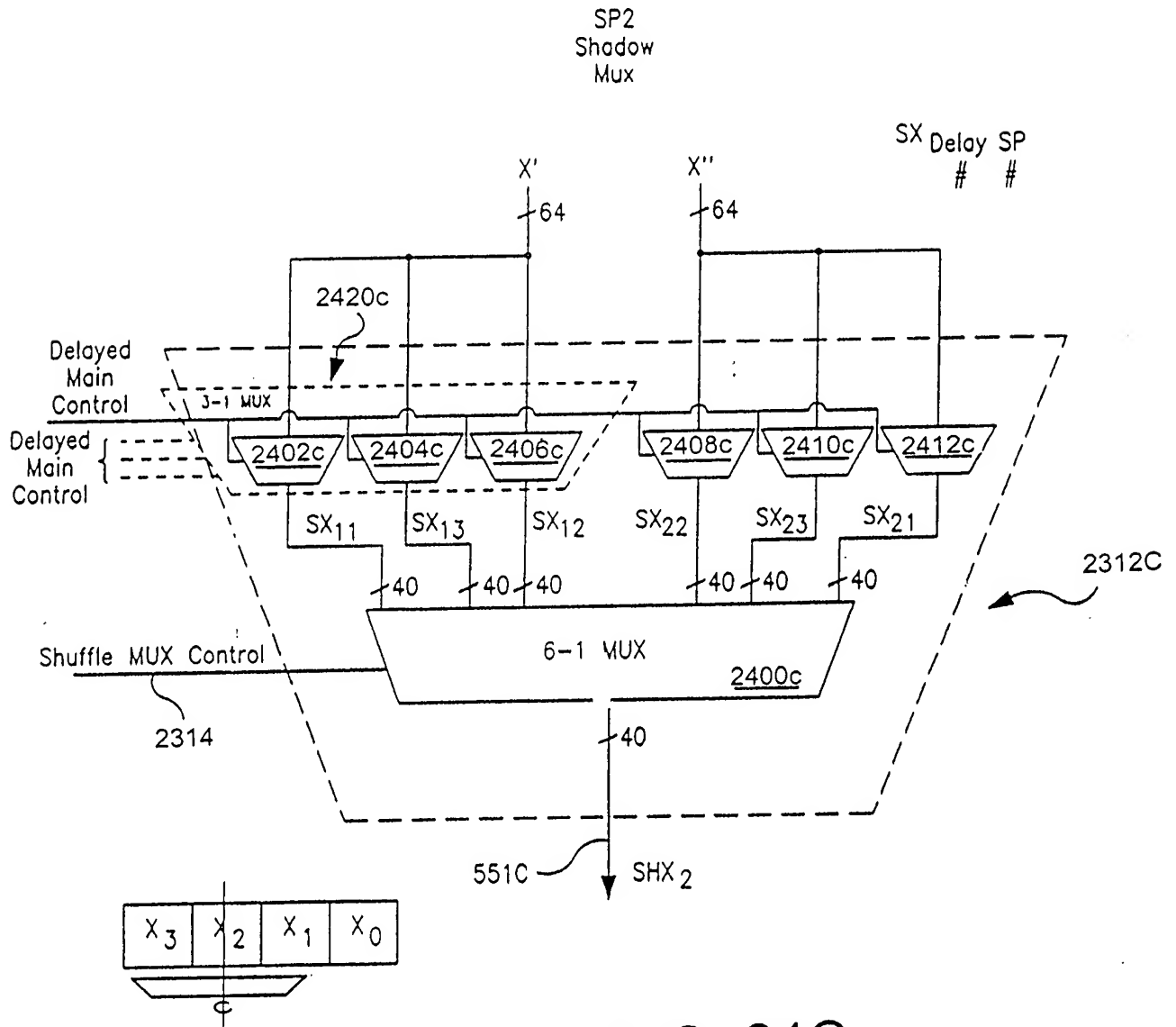


FIG. 24B



$$\begin{aligned} X_2 &= SX_{12}, SX_{22} \\ X_3 &= SX_{13}, SX_{23} \\ X_1 &= SX_{11}, SX_{21} \end{aligned}$$

FIG. 24C

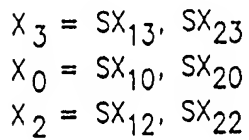


FIG. 24D

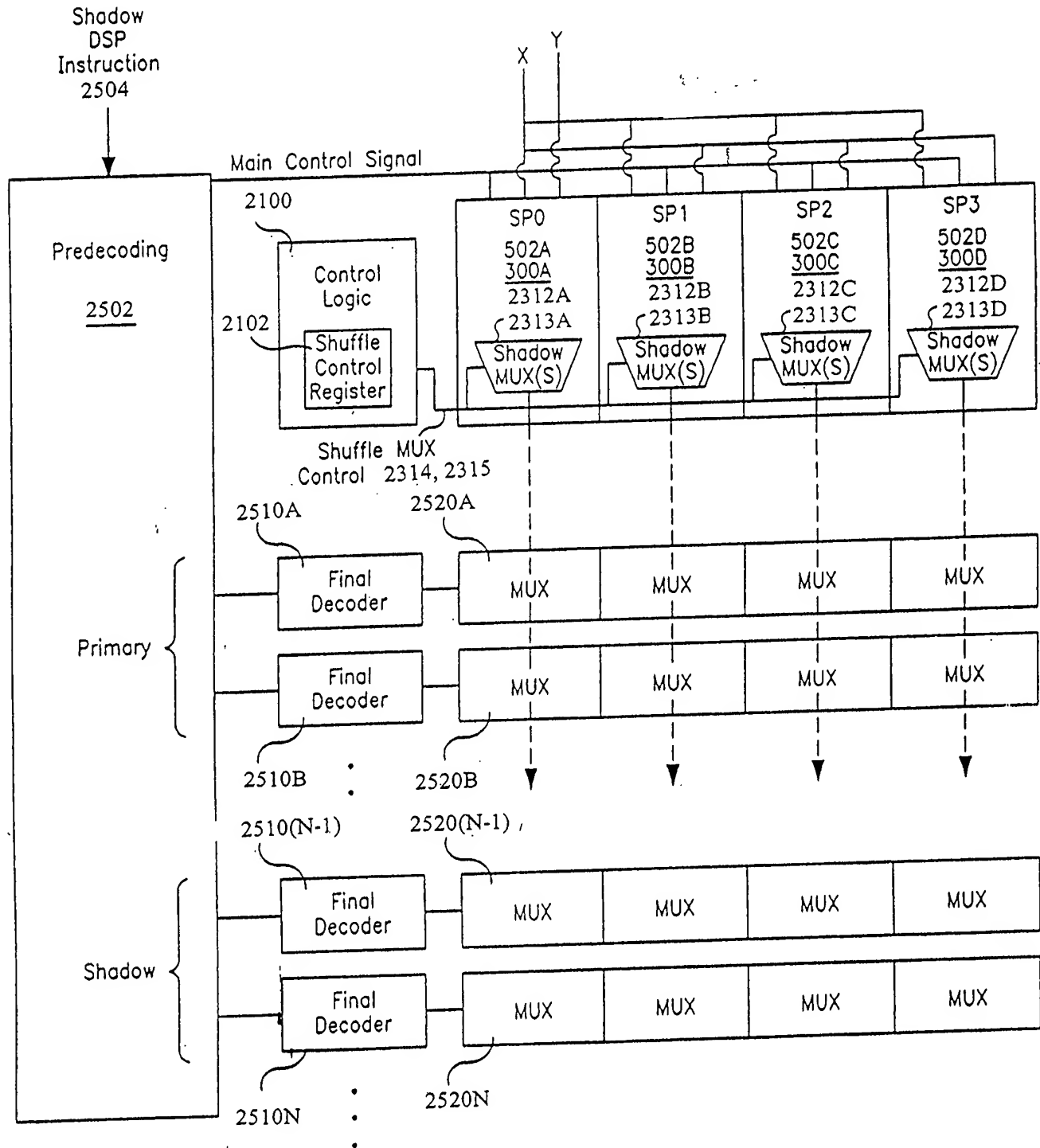


FIG. 25

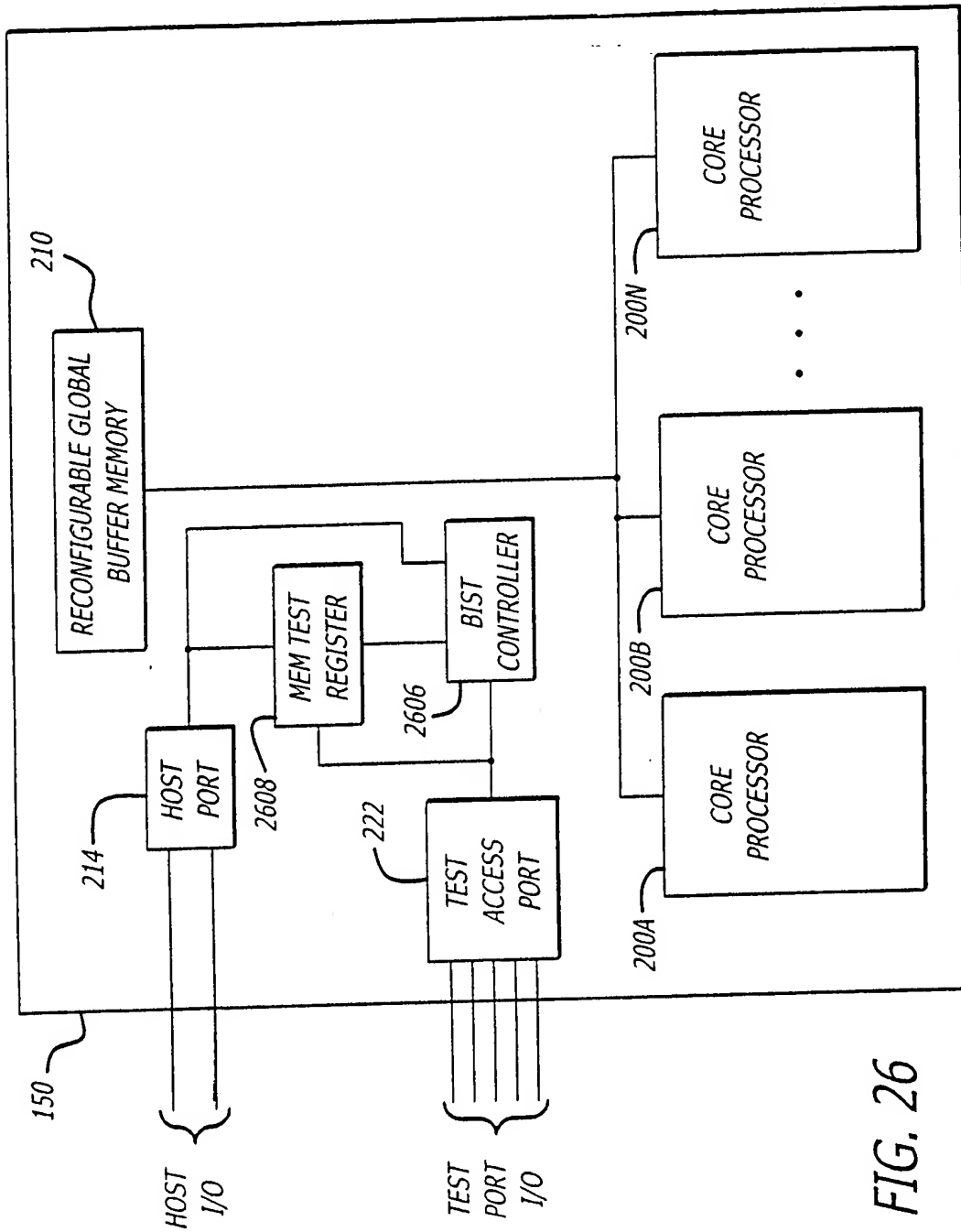


FIG. 26

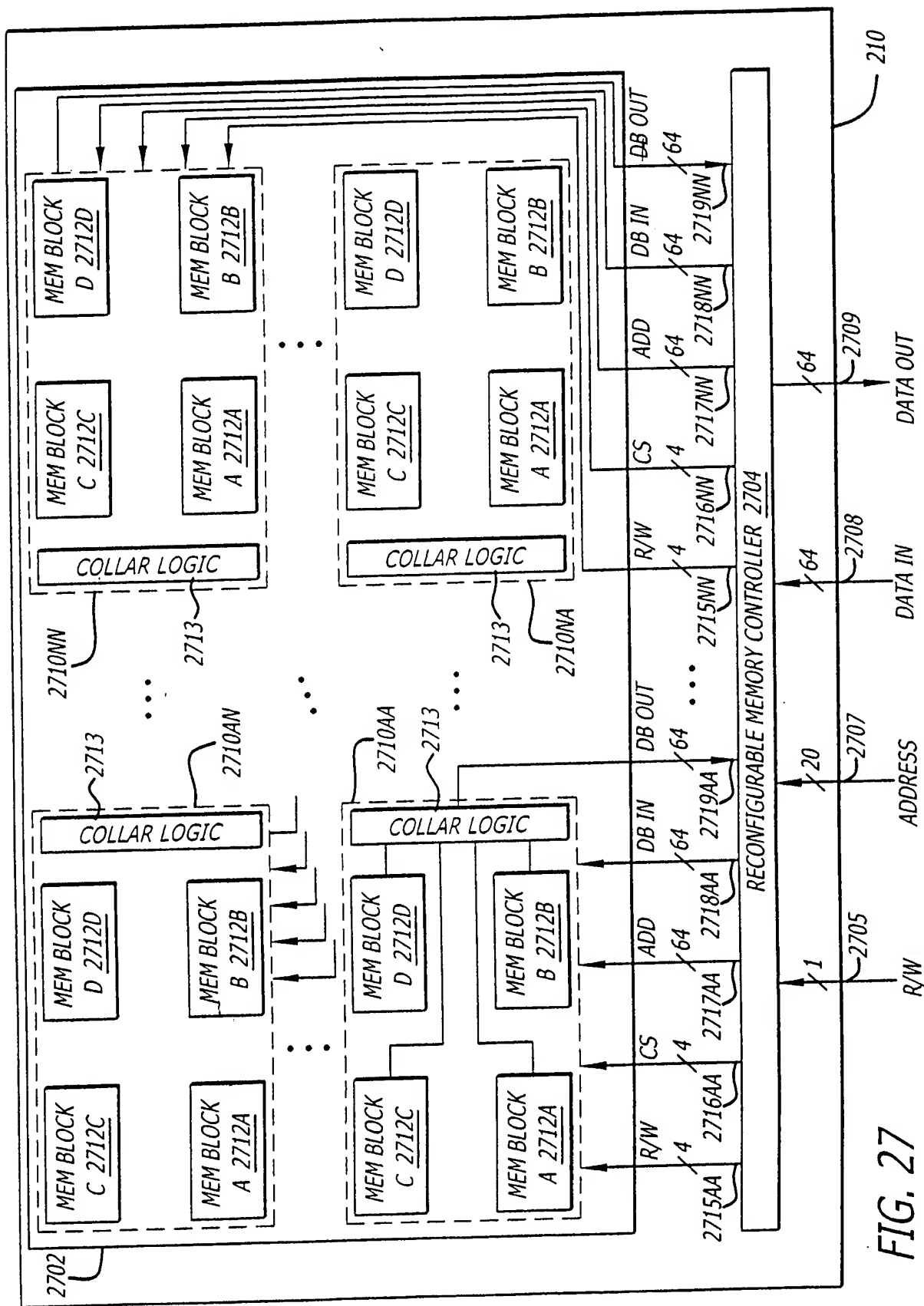


FIG. 27

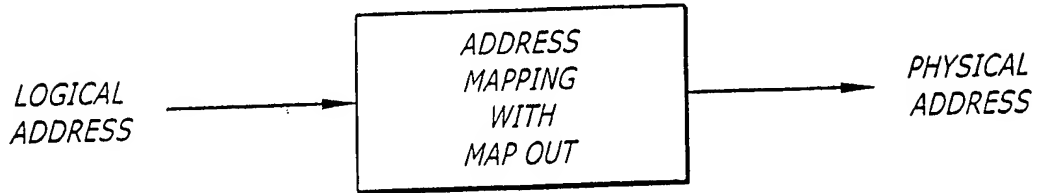


FIG. 28

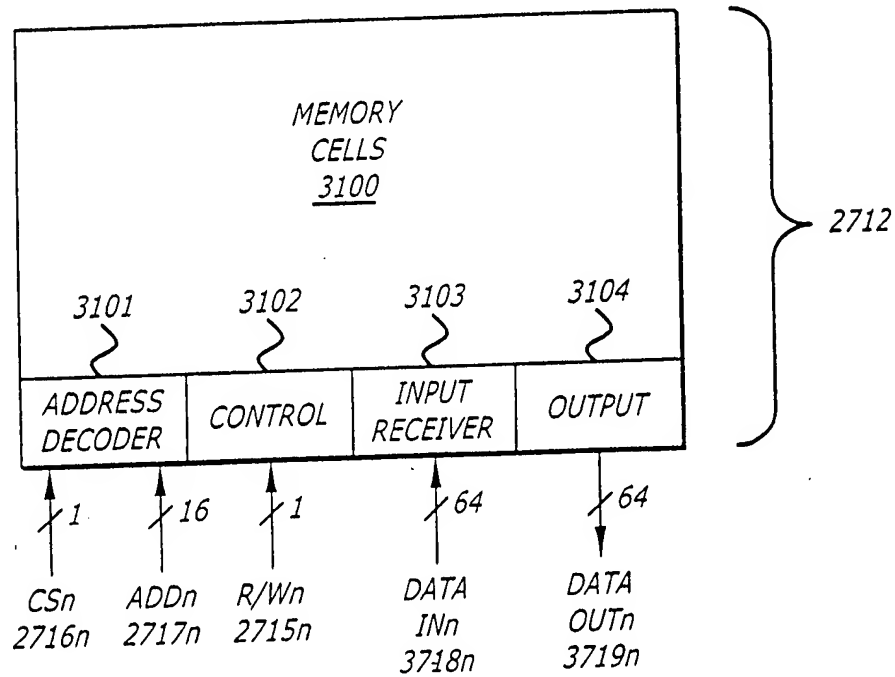


FIG. 31

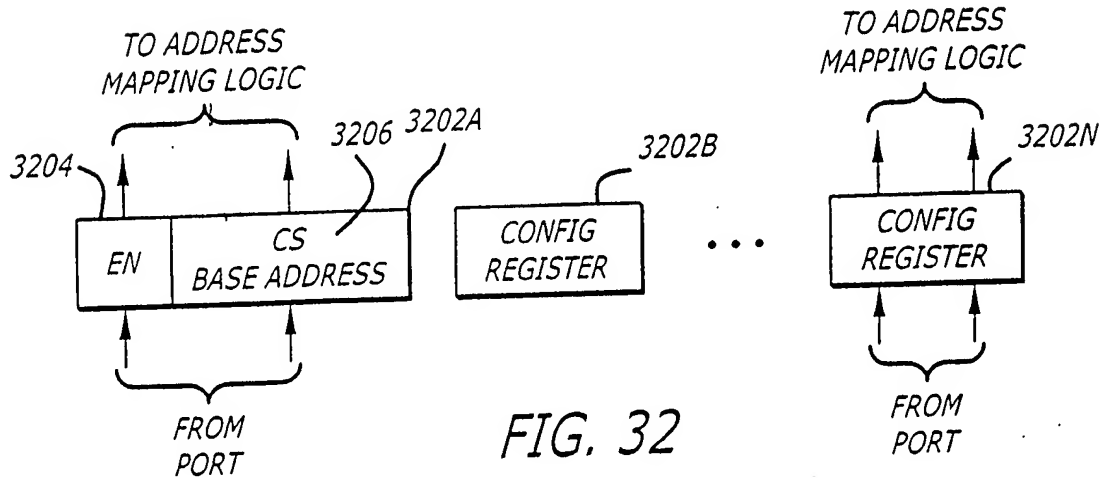
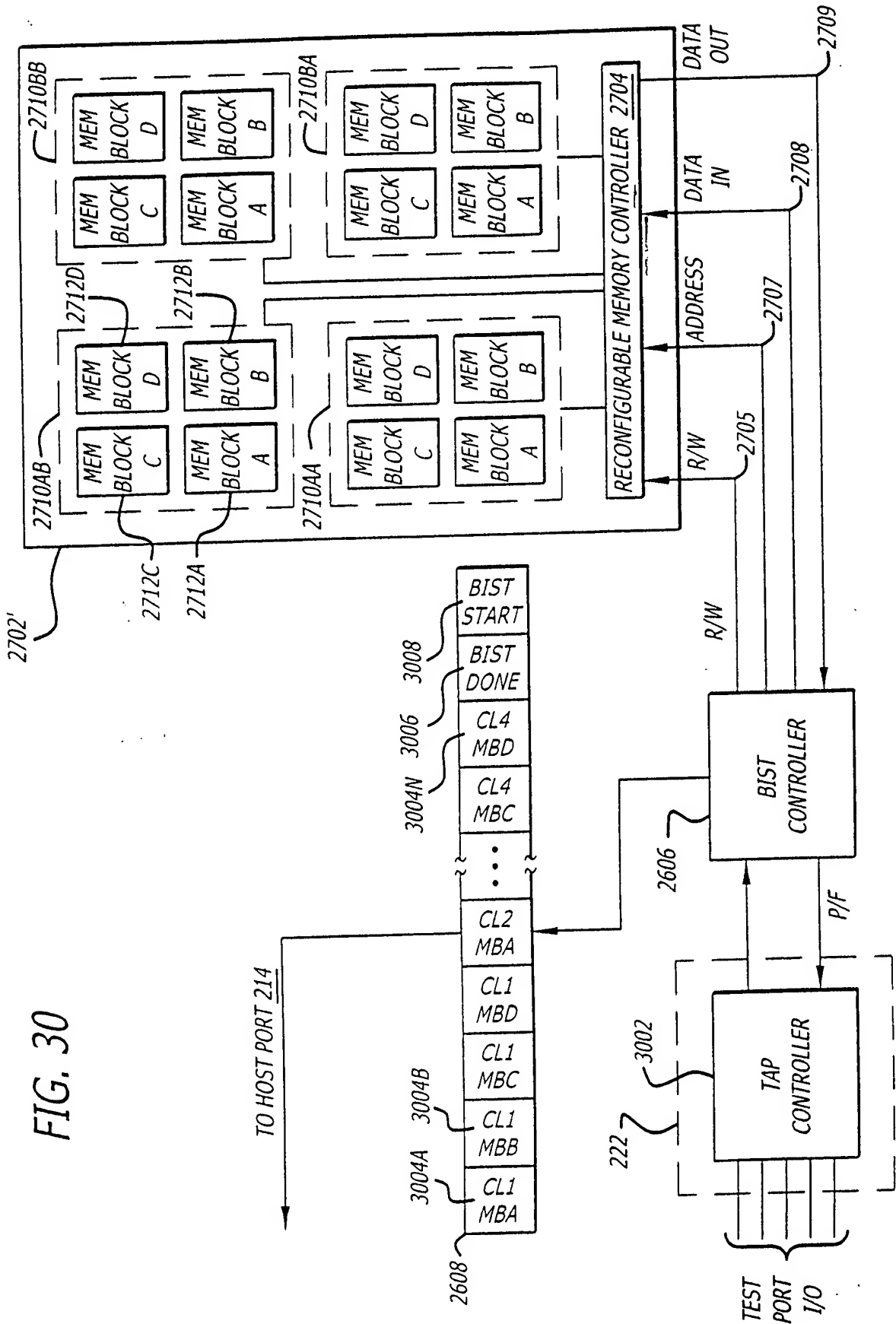
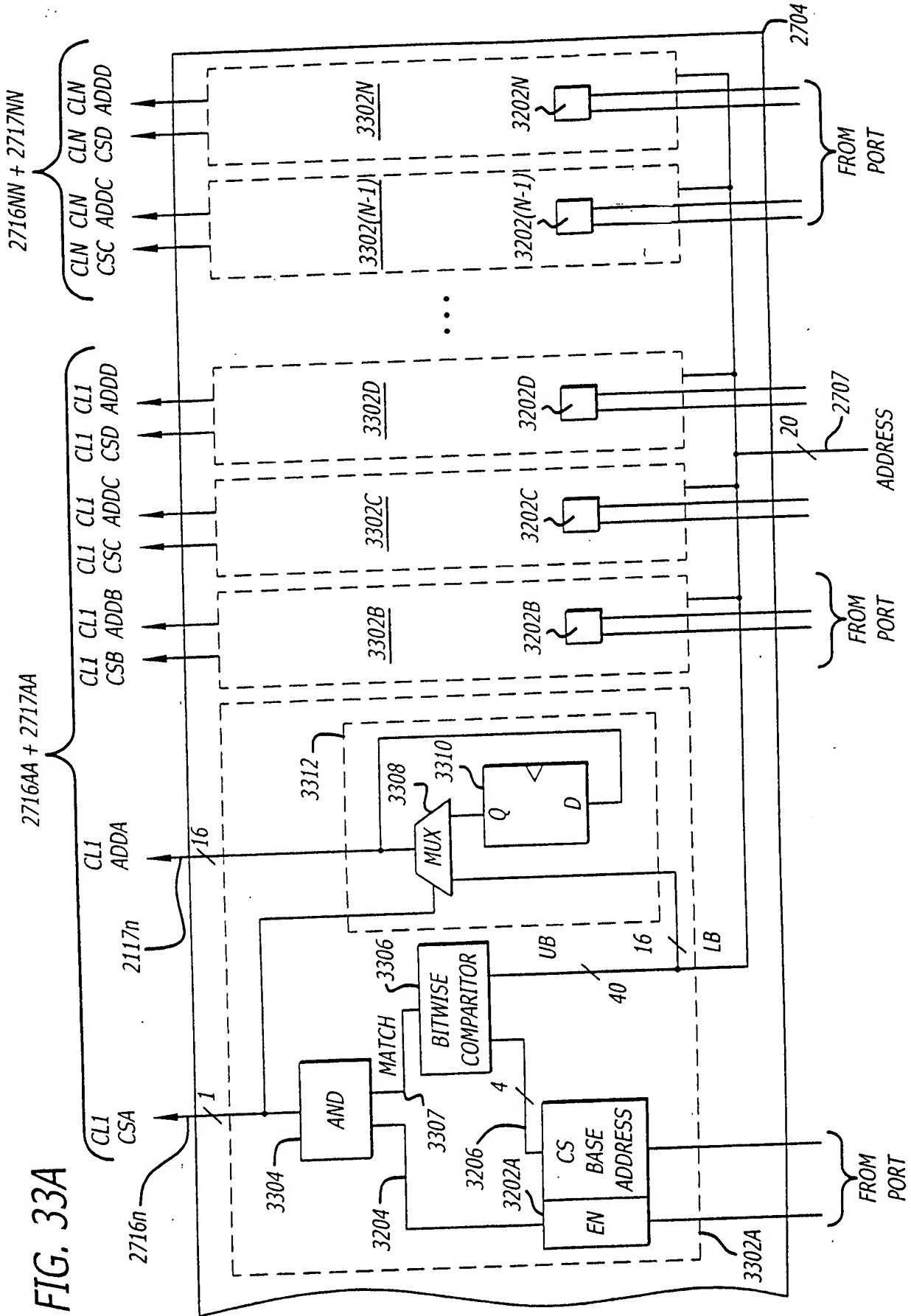


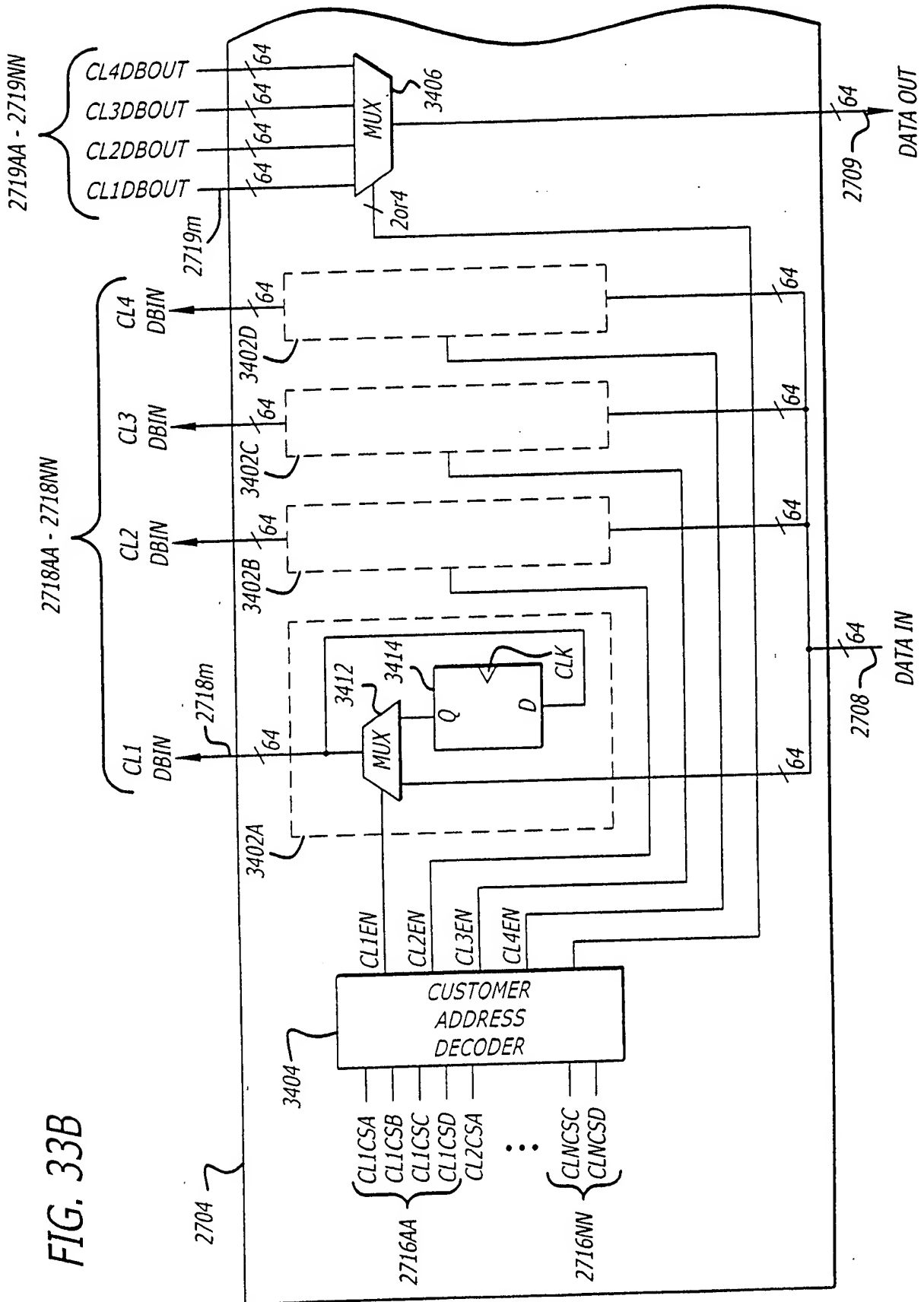
FIG. 32

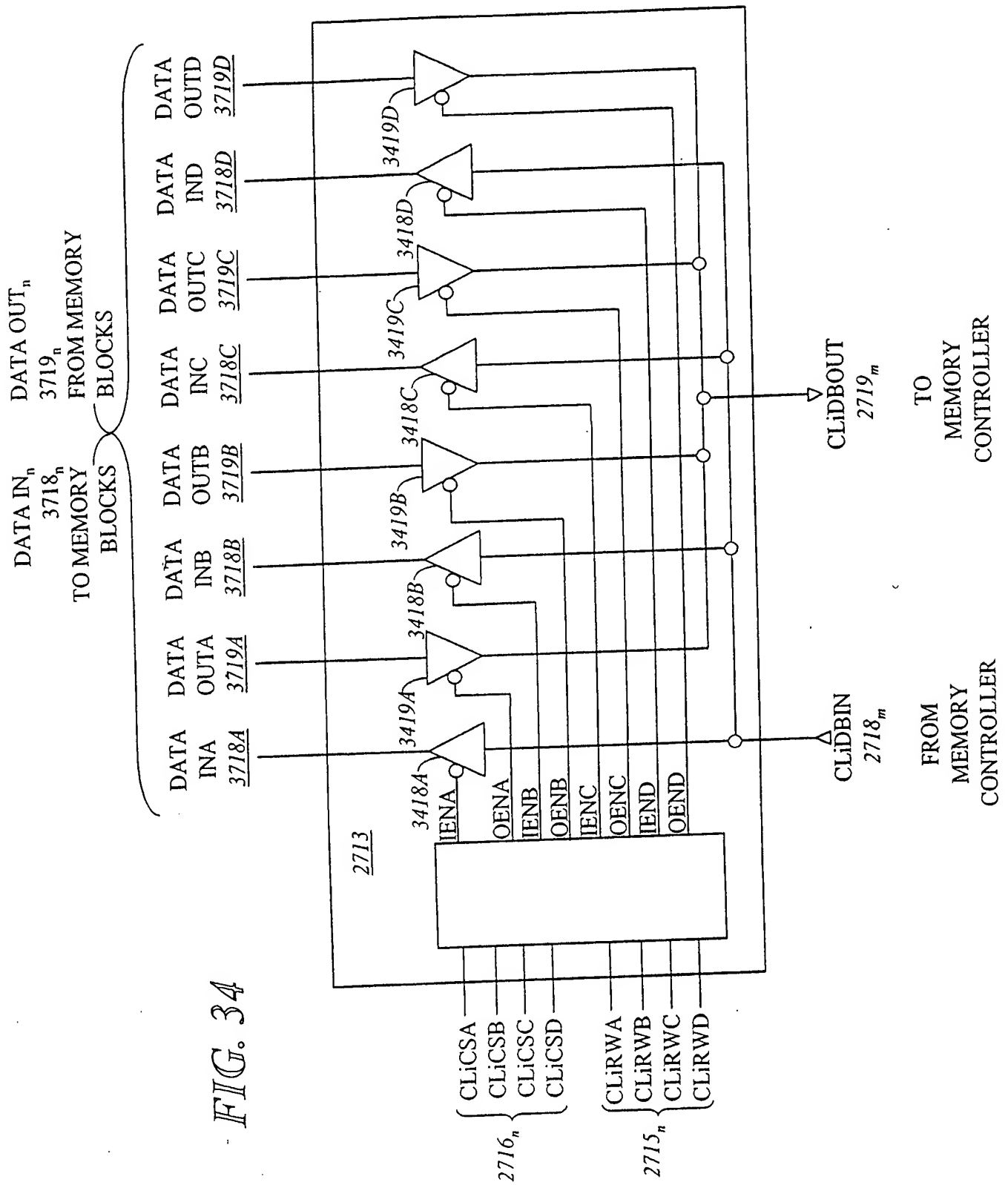
FIG. 29

LOGICAL ADDRESS (# WORDS)	LOGICAL BITS	ASSUME 8 BITS/WORD	PHYSICAL BITS	PHYSICAL ADDRESS (# WORDS)
MAX/8 - MOA = MAX/8 - 64K	↔	MEM BLOCK DN	MAX	MAX/8
MAX/8 - 128K	↔	MEM BLOCK CN	MAX - 512K	MAX/8 - 64K
MAX/8 - 192K	↔	MEM BLOCK BN	MAX - 1024K	MAX/8 - 128K
MAX/8 - 256K	↔	MEM BLOCK AN	MAX - 1536K	MAX/8 - 192K
MAX/8 - 320K	↔		MAX - 2048K	MAX/8 - 256K
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
448K	↔	MEM BLOCK D2	4096K	512K
384K	↔	MEM BLOCK C2	3548K	448K
320K	↔	MEM BLOCK B2	3072K	384K
256K	↔	MEM BLOCK A2	2560K	320K
192K	↔		2048K	256K
(192K - 1)	↔	MEM BLOCK D1 2712D/	(2048K - 1)	(256K - 1)
128K	↔	MEM BLOCK C1 2712C	1536K	192K
64K	↔	MEM BLOCK B1 2712B	1024K	128K
0K	↔	MEM BLOCK A1 2712A	512K	64K
	↔		0K	0K



[illegible]





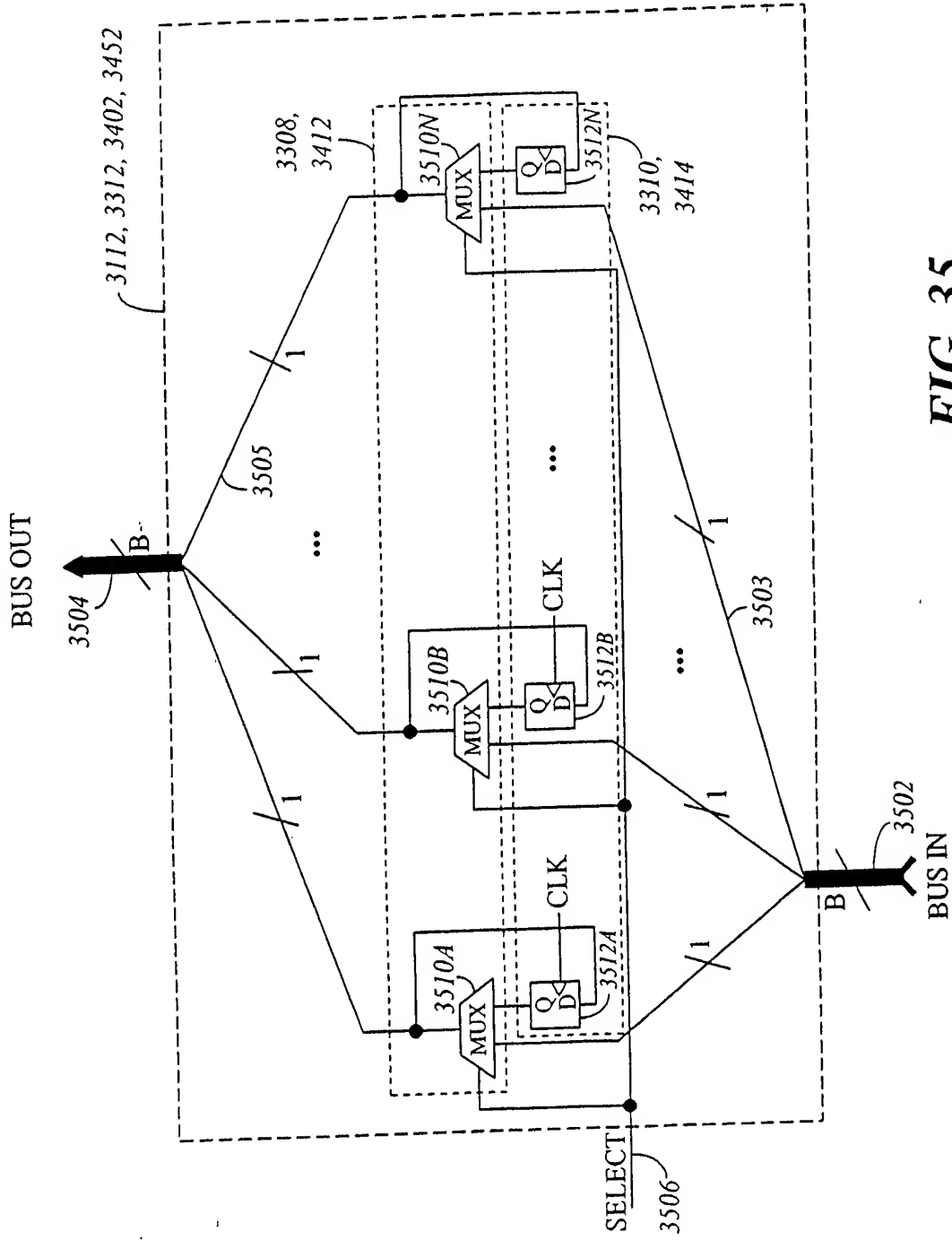


FIG. 35

202

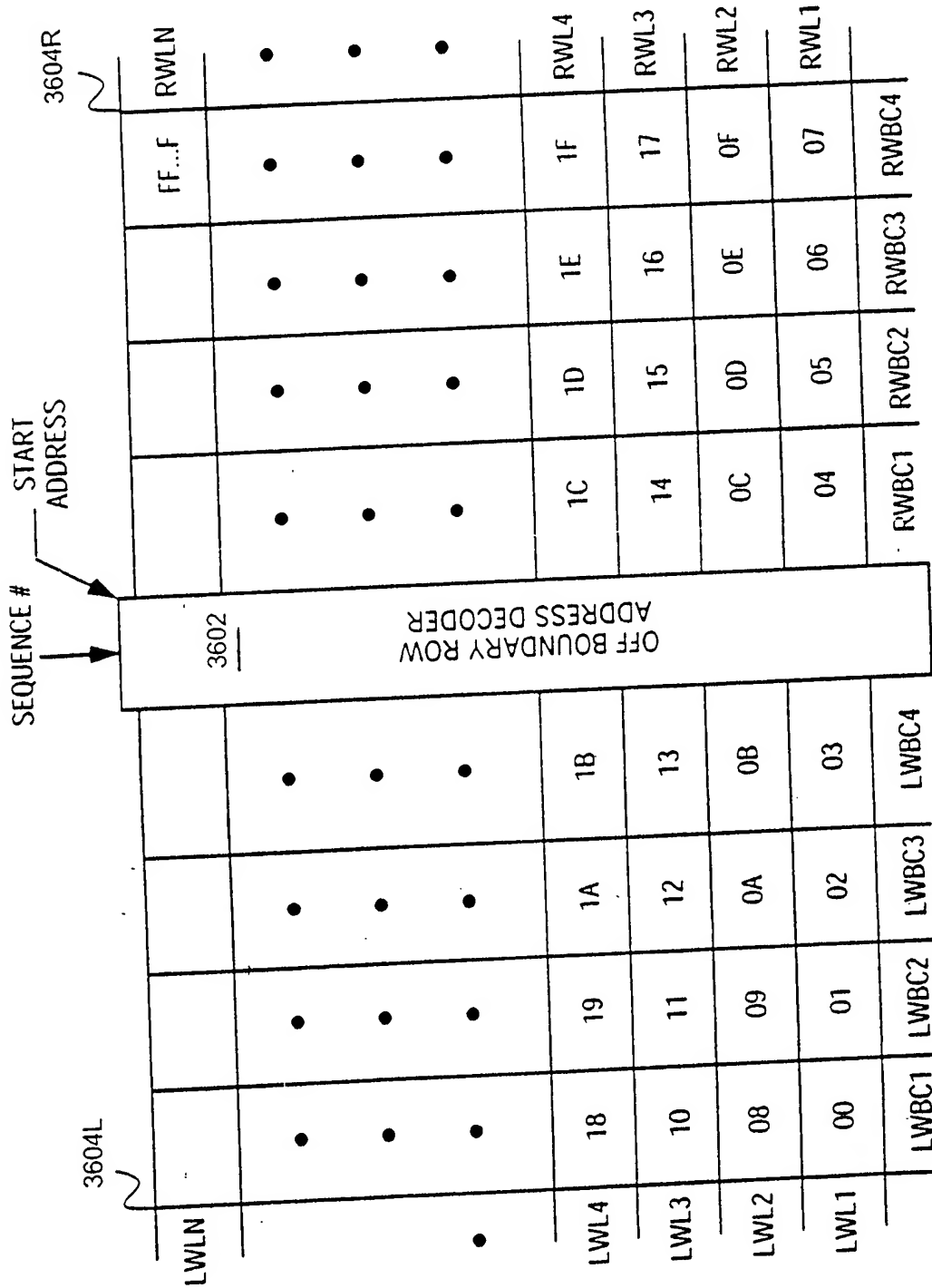


FIG. 36A

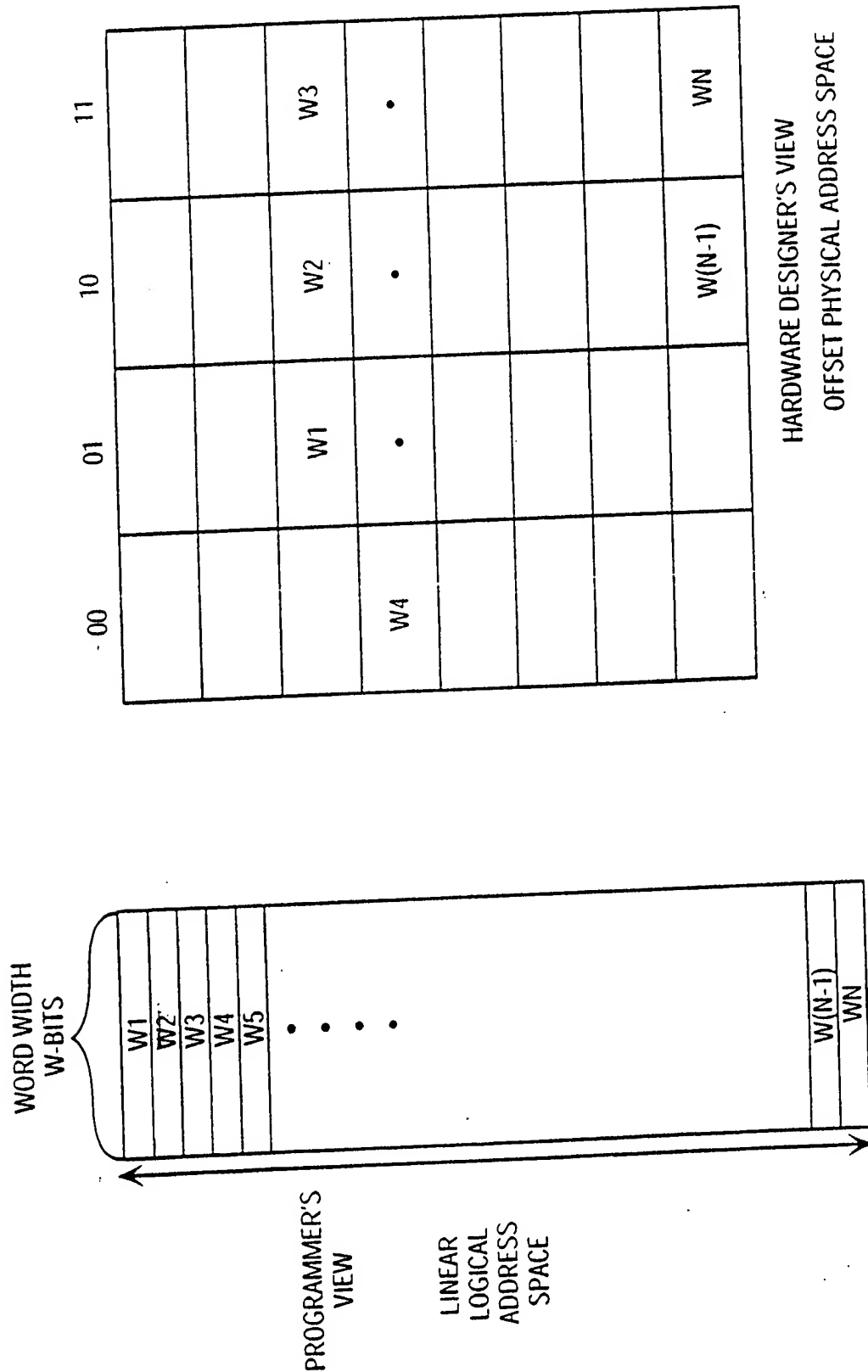


FIG. 36B

FIG. 36C

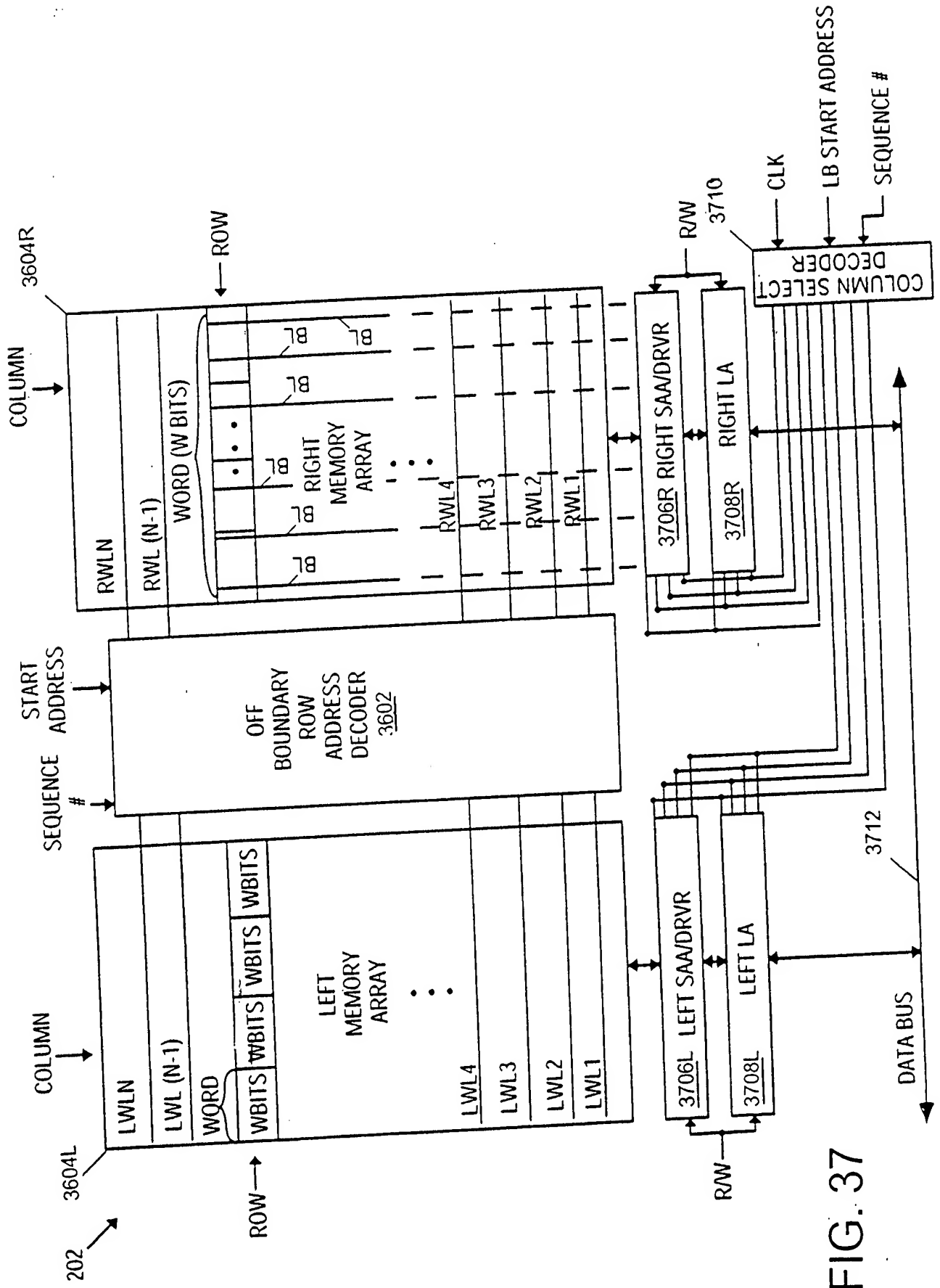


FIG. 37

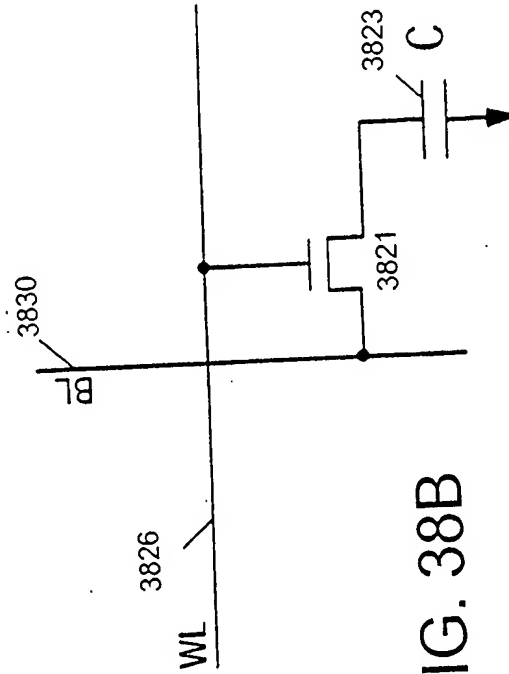


FIG. 38B

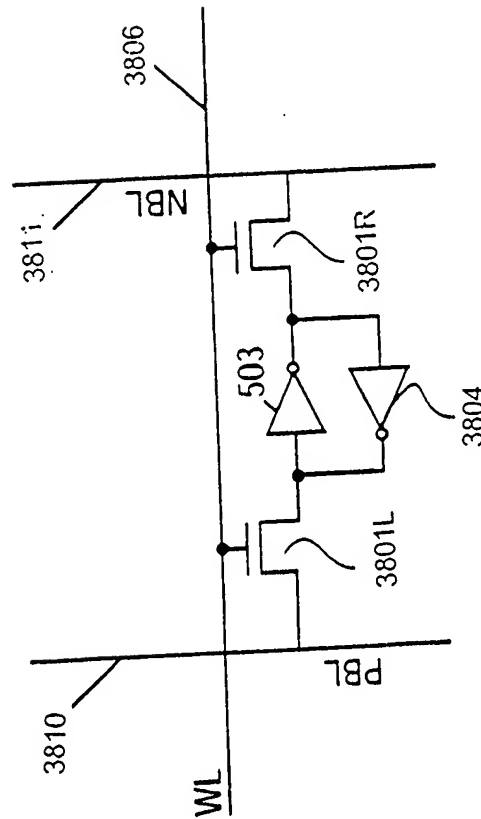
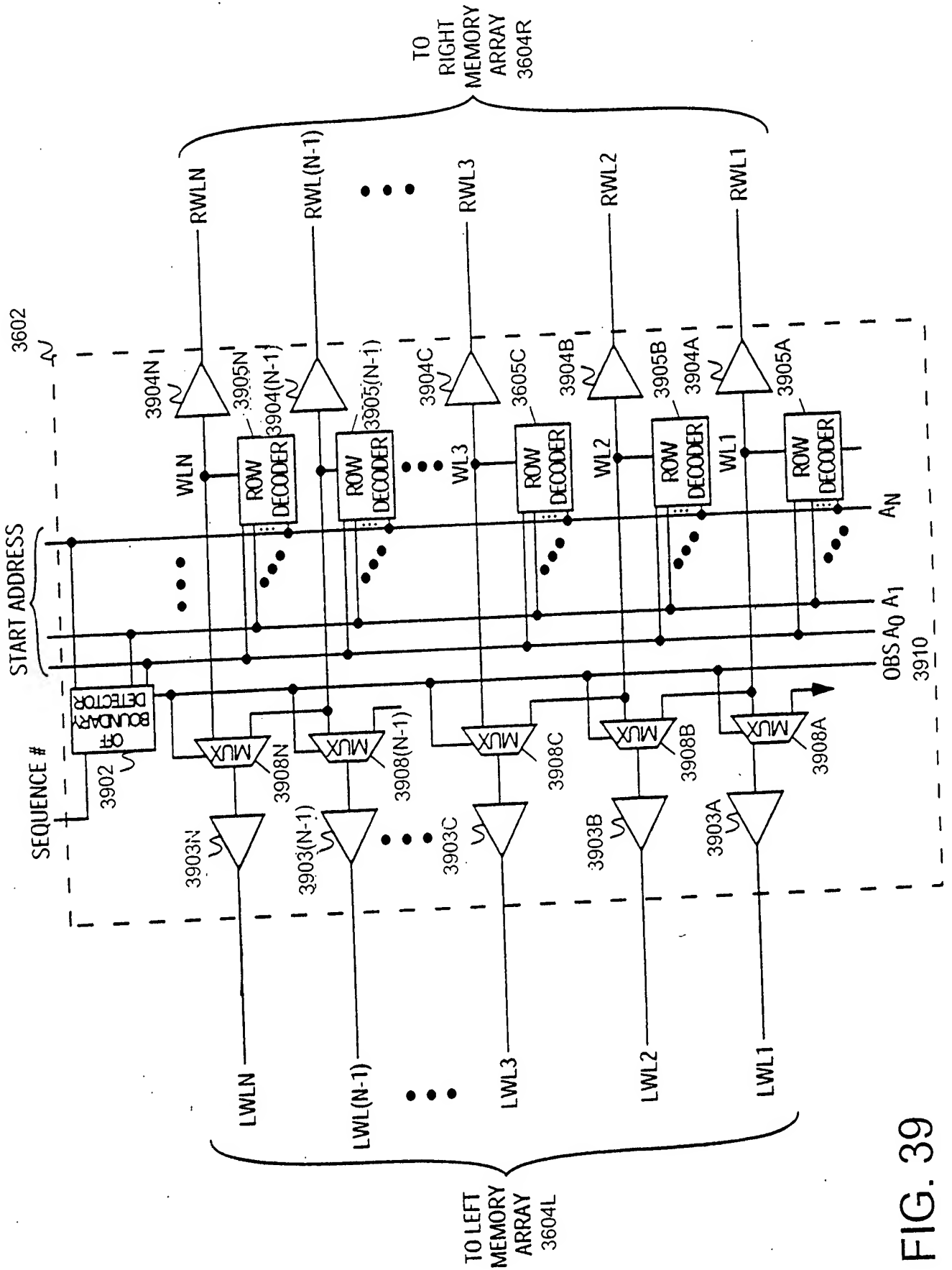


FIG. 38A



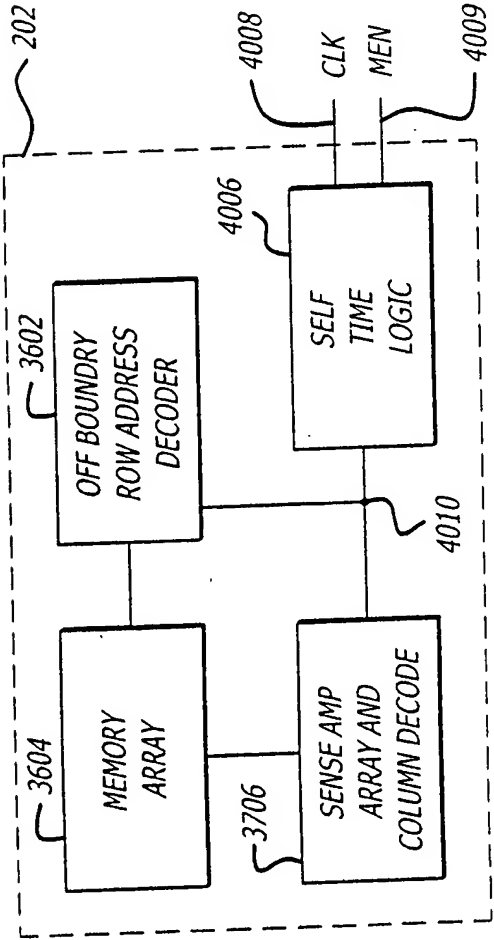


FIG. 40

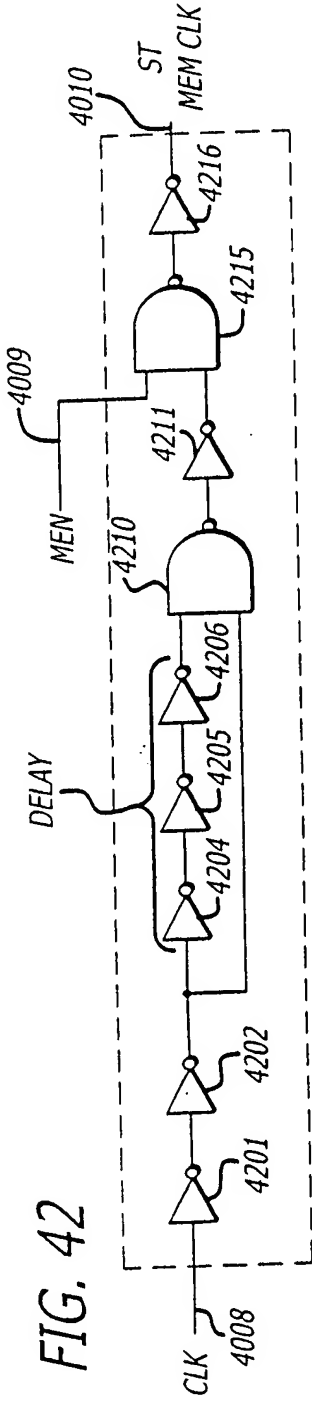


FIG. 42

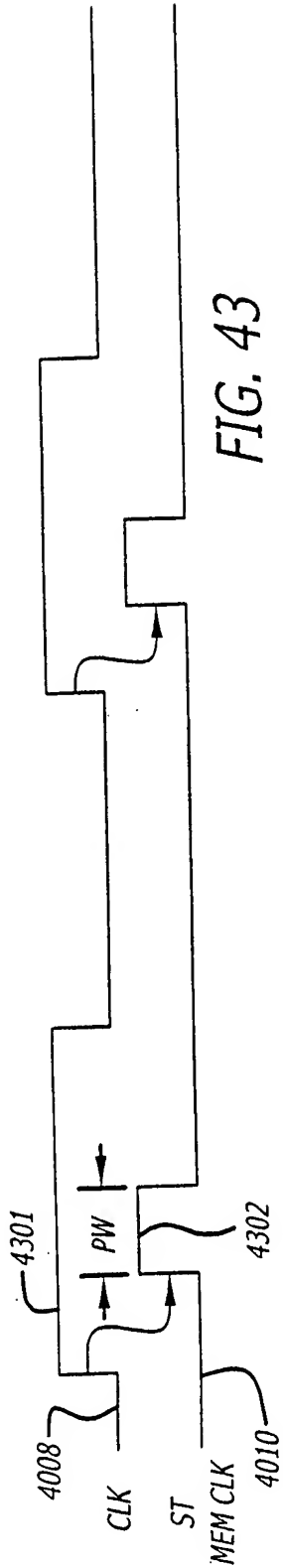
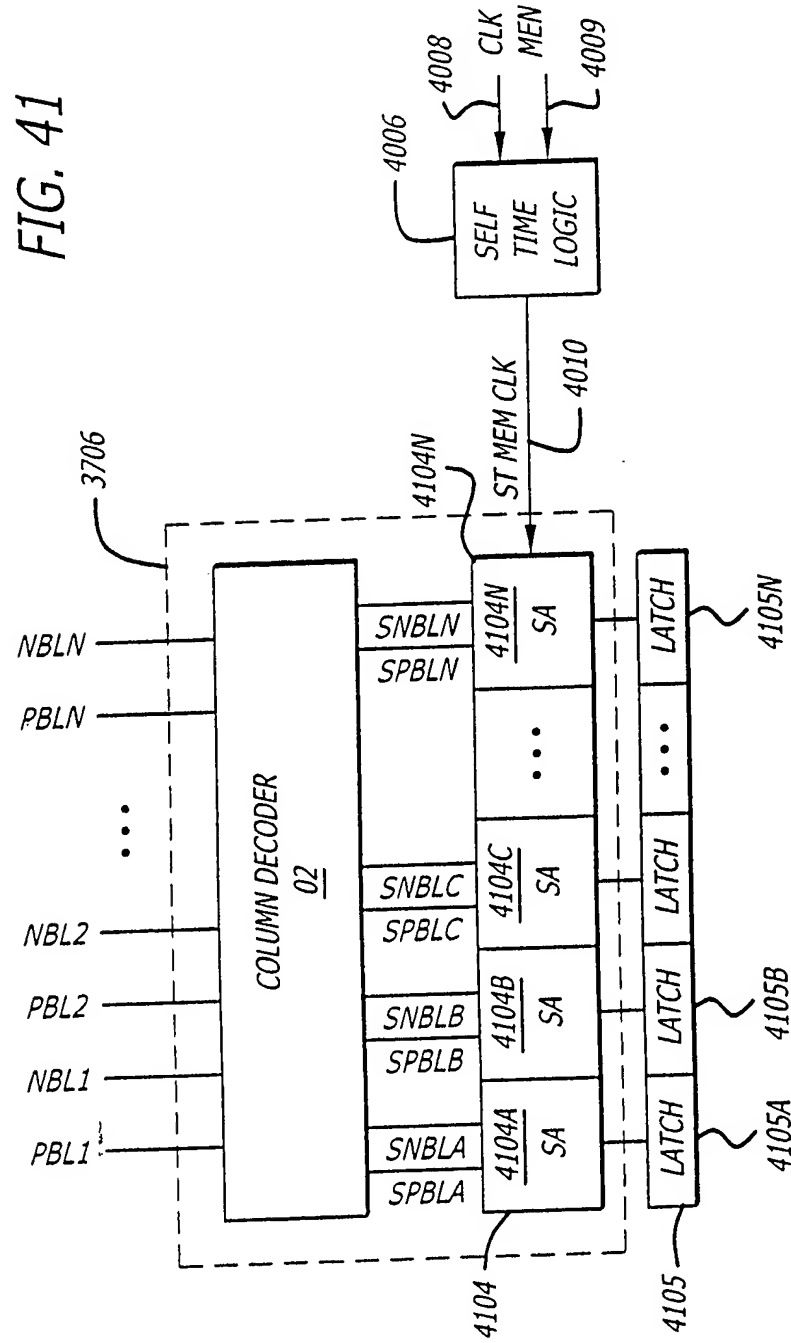


FIG. 43

FIG. 41



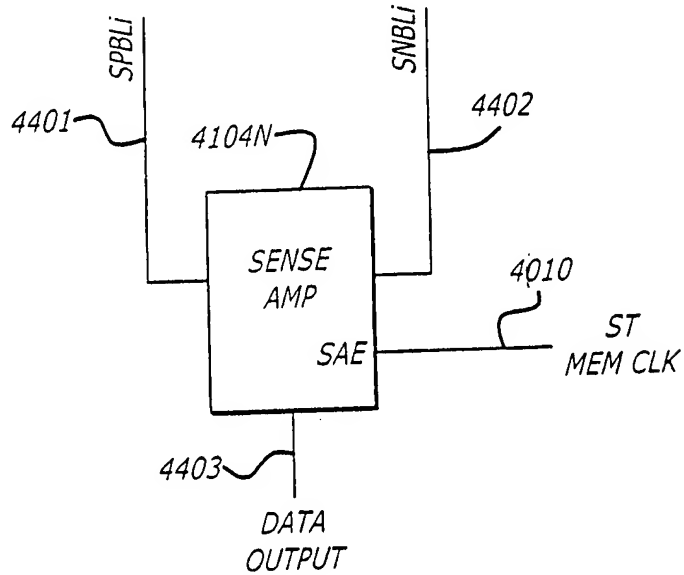


FIG. 44A

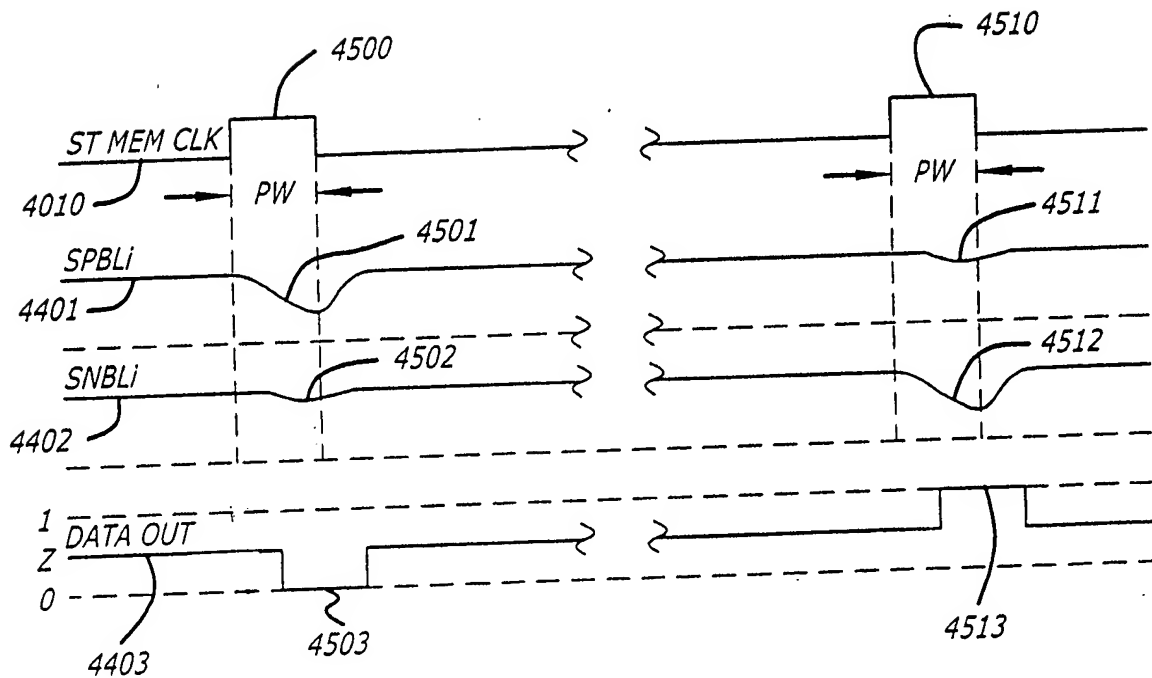
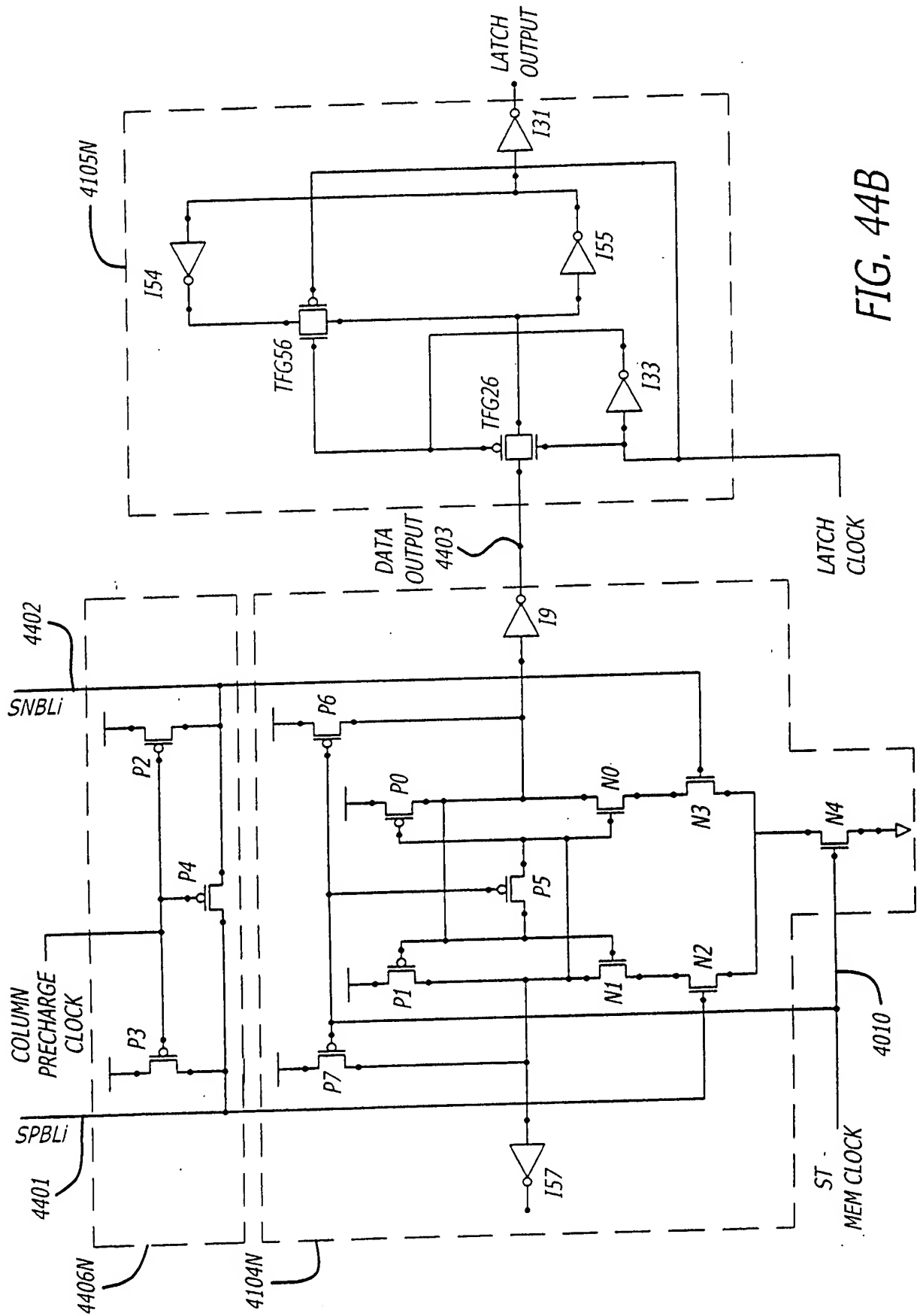


FIG. 45



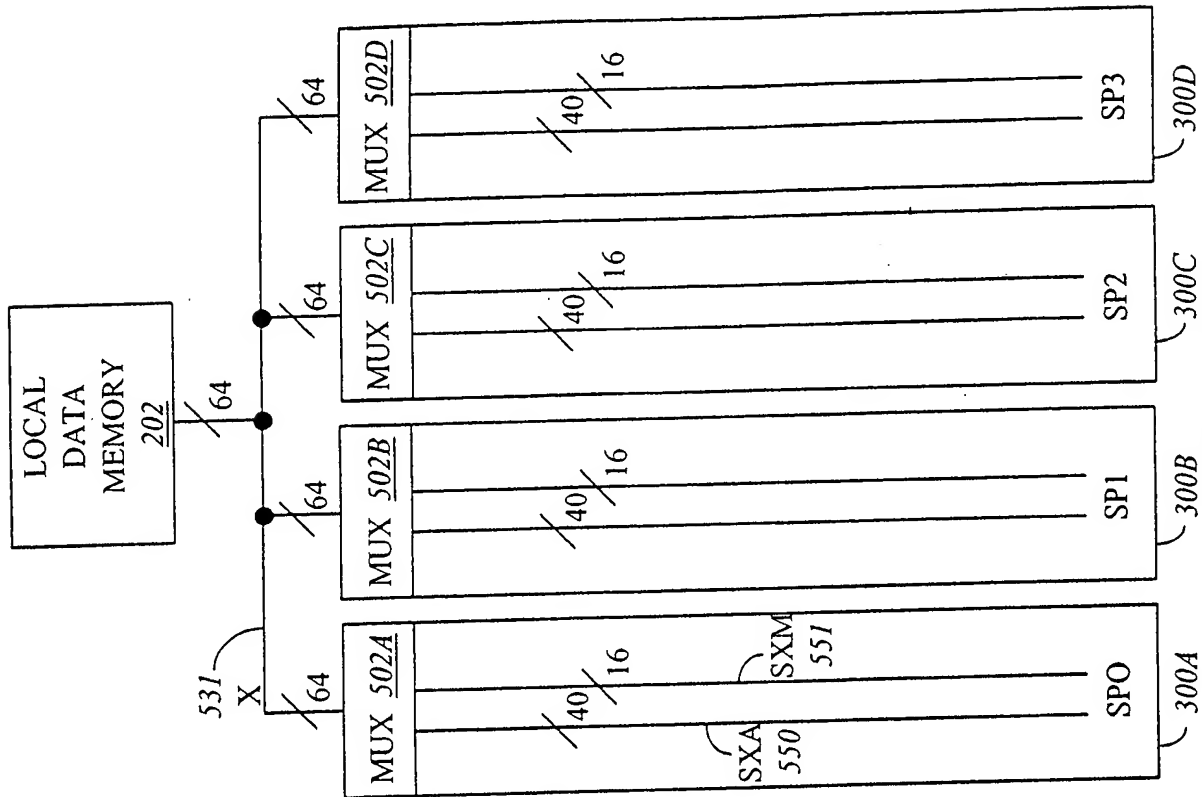


FIG. 46B

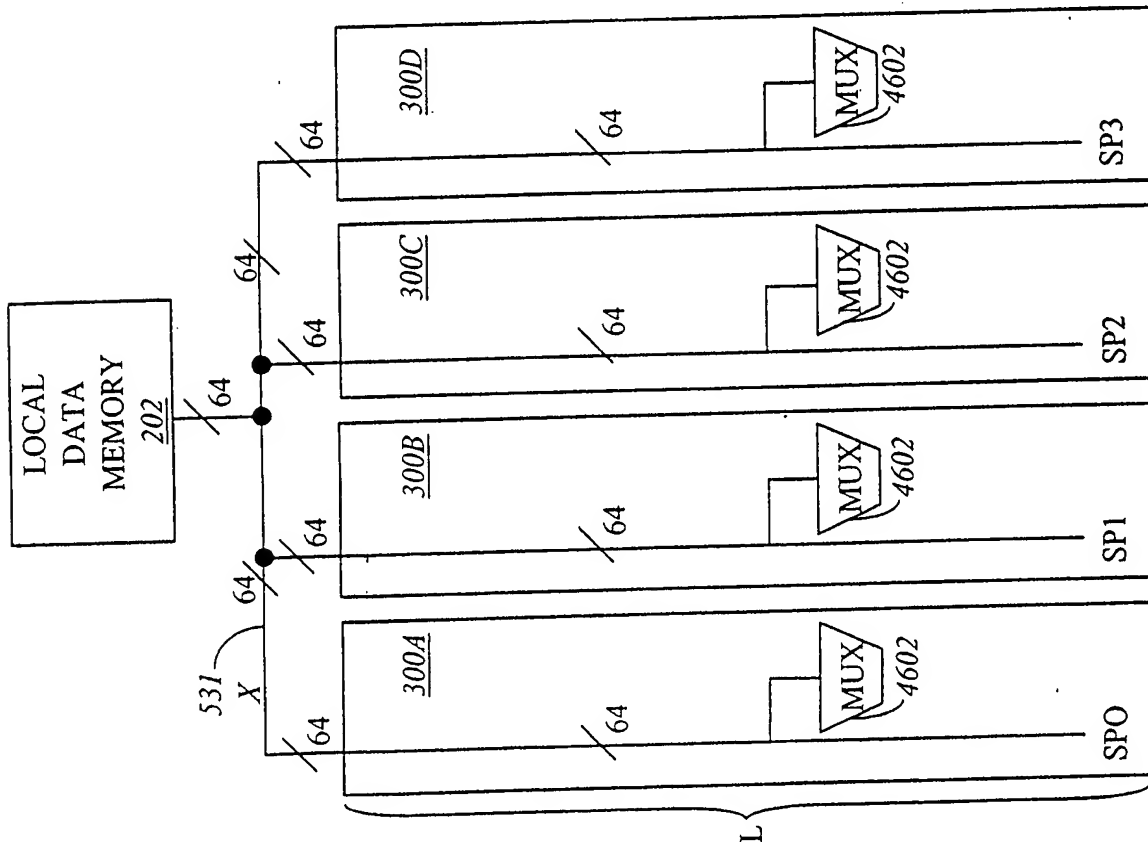


FIG. 46A